PA01-308

reference 4

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平10－340276

(43)公開日　平成10年(1998)12月22日

(51)Int.Cl.[6]	識別記号	FI	
G06F 17/30		G06F 15/403	340D
17/18		15/38	Z
		15/40	310C
			380D

審査請求 有　請求項の数18 OL （全 29 頁）

(21)出願番号　特願平9－339185

(22)出願日　平成9年(1997)12月9日

(31)優先権主張番号　特願平9－90384

(32)優先日　平9(1997)4月9日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　592073101
日本アイ・ビー・エム株式会社
東京都港区六本木３丁目２番12号

(72)発明者　松澤　裕史
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社東京基礎研究所内

(72)発明者　福田　剛志
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社東京基礎研究所内

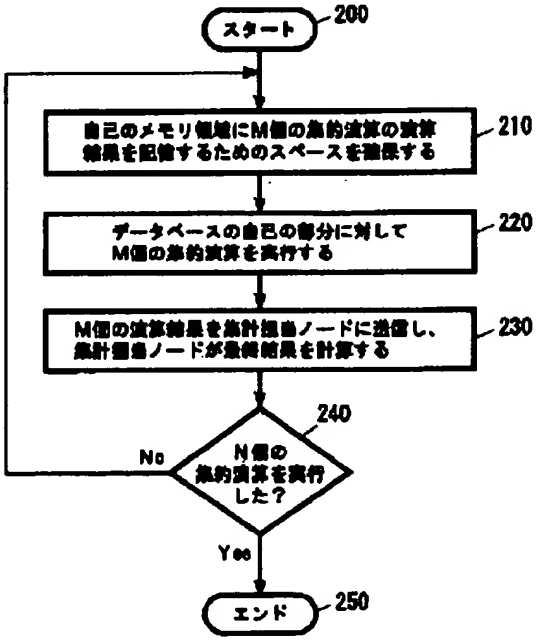
(74)代理人　弁理士　坂口　博　（外１名）

(54)【発明の名称】　集約演算実行方法及びコンピュータ・システム

(57)【要約】

【課題】複数の集約演算を並列に高速実行する方法を提供すること

【解決手段】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムにおいて、（ａ）自己のメモリ領域に、Ｎ個の集約演算のうちＭ個（Ｎ以下の整数）分の演算結果を記憶するスペースを確保するステップと、（ｂ）Ｍ個の集約演算をまとめて、データベースの自己の部分に対して実行するステップと、（ｃ）Ｍ個の集約演算の各プロセッサにおける演算結果を、集計すべきプロセッサに送信し、集計すべきプロセッサが最終結果を計算するステップと、（ｄ）Ｎ個の集約演算が終了するまで、ステップ（ａ）乃至（ｃ）を繰り返すステップとを含む。



【特許請求の範囲】

【請求項１】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムにおいて、前記グループ毎に集約を行う演算を含む、Ｎ個の集約演算を前記データベースに対し実行する方法であって、（ａ）各前記プロセッサが、前記自己のメモリ領域に、前記Ｎ個の集約演算のうちＭ個（Ｎ以下の整数）分の演算結果を記憶するスペースを確保するステップと、（ｂ）各前記プロセッサが、Ｍ個の集約演算をまとめて、前記データベースの自己の部分に対して実行するステップと、（ｃ）各前記プロセッサが、前記Ｍ個の集約演算の各プロセッサにおける演算結果を、集計すべきプロセッサに送信し、前記集計すべきプロセッサが最終結果を計算するステップと、（ｄ）前記Ｎ個の集約演算の実行が終了するまで、前記ステップ（ａ）乃至（ｃ）を繰り返すステップとを含む集約演算実行方法。

【請求項２】前記ステップ（ｂ）が、（b1）各前記プロセッサが、前記自己のメモリ領域のワークスペースに、前記データベースの自己の部分である部分データベースの一部を読み出すステップと、（b2）各前記プロセッサが、前記自己のメモリ領域に格納された本ステップ以前の演算結果と読み出された前記部分データベースの一部とについて、前記Ｍ個の集約演算を実行するステップと、（b3）前記部分データベースのすべてに対して前記Ｍ個の集約演算が実行されるまで、前記ステップ(b1)及び(b2)を繰り返すステップとを含む請求項１記載の集約演算実行方法。

【請求項３】前記Ｍは集約演算の演算結果のために前記自己のメモリ領域に確保できるスペースから決定されることを特徴とする請求項１記載の集約演算実行方法。

【請求項４】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムにおいて、前記グループ毎に集約を行う演算を含む、Ｐ個の集約演算を前記データベースに対し実行する方法であって、（ａ）各前記プロセッサが、前記Ｐ個の集約演算のうち自らが実行するＱ個の集約演算の演算結果を記憶するスペースを、前記自己のメモリ領域に確保するステップと、（ｂ）各前記プロセッサが、前記自己のメモリ領域に前記データベースの自己の部分である部分データベースの一部を読み出し、読み出された前記部分データベースの一部を前記ネットワークを介してブロードキャストすることを繰り返して、各前記プロセッサが全ての前記データベースのデータに対し前記自らが実行するＱ個の集約演算を実行するステップと、（ｃ）前記Ｐ個の集約演算が実行されるまで、前記ステップ（ａ）及び（ｂ）を繰り返すステップとを含む集約演算実行方法。

【請求項５】前記ステップ（ａ）が、（a1）１つの集約演算の演算結果を格納できるようなスペースが１つのプロセッサの前記自己のメモリ領域に存在するか検査するステップと、（a2）前記スペースが存在する場合には、前記スペースを前記１つの集約演算の演算結果のために確保するステップと、（a3）前記スペースが存在しない場合には、他のプロセッサの前記メモリ領域に前記１つの集約演算の演算結果を格納できるようなスペースが存在するか検査するステップと、（a4）前記他のプロセッサの前記メモリ領域にスペースが存在する場合には、当該他のプロセッサのメモリ領域のスペースを前記１つの集約演算の演算結果のために確保するステップとを含み、

前記他のプロセッサのメモリ領域にスペースが存在しない場合には、後の繰り返し処理にて前記１つの集約演算を実行することを特徴とする請求項４記載の集約演算実行方法。

【請求項６】前記Ｑは集約演算の演算結果のために前記自己のメモリ領域に確保できるスペースから決定されることを特徴とする請求項４記載の集約演算実行方法。

【請求項７】前記ステップ（ｂ）が、（b1）各前記プロセッサが、前記自己のメモリ領域のワークスペースに、前記部分データベースの一部を読み出すステップと、（b2）読み出された前記部分データベースの一部を前記ネットワークを介してブロードキャストするステップと、（b3）前記自己のメモリ領域に格納された本ステップ以前の演算結果と前記読み出された部分データベースの一部と他のプロセッサから送られてきたデータとについて、前記自らが実行するＱ個の集約演算を実行するステップと、（b4）各前記プロセッサが全ての前記データベースの内容に対し前記自らが実行するＱ個の集約演算を実行するまで前記ステップ（b1）乃至（b3）を繰り返すステップとを含む請求項４記載の集約演算実行方法。

【請求項８】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムにおいて、前記グループ毎に集約を行う演算を含む、Ｓ個の集約演算を前記データベースに対し実行する方法であって、（ａ）前記Ｓ個の集約演算のうち実行するＴ個の集約演算を決定するステップと、（ｂ）前記Ｔ個の集約演算を実行する際に集約される、各集約演算の各グループを取り扱うプロセッサを決定するステップと、（ｃ）各前記プロセッサが、前記データベースの自己の部分である部分データベースの一部を前記自己のメモリ領域に読み出し、読み出されたデータのうち他のプロセッサが集約すべき集約演算のグループに関するデータを当該集約演算のＩＤと共に前記他のプロセッサに前記ネットワークを介して送信し、自己が集約すべき集約

演算のグループに関するデータについて前記Ｔ個の集約演算を実行するステップと、（ｄ）各前記プロセッサが集約すべき集約演算のグループに関する全てのデータに対して前記Ｔ個の集約演算を実行するまで、前記ステップ（ｃ）を実行するステップと、（ｅ）前記Ｓ個の集約演算を実行するまで、前記ステップ（ａ）乃至（ｄ）を繰り返すステップとを含む集約演算実行方法。

【請求項９】前記ステップ（ｃ）が、（ｃ1）各前記プロセッサが、前記データベースの自己の部分である部分データベースの一部を前記自己のメモリ領域のワークスペースに読み出すステップと、（ｃ2）各前記プロセッサが、読み出されたデータの各部分を必要とするプロセッサを求め、当該部分を、関連する集約演算のＩＤと共に前記必要とするプロセッサに前記ネットワークを介して送信するステップと、（ｃ3）各前記プロセッサが、前記読み出されたデータのうち自己が集約すべき集約演算のグループに関するデータ及び前記他のプロセッサからのデータ及び前記自己のメモリ領域に格納された本ステップ以前の演算結果に対し、前記Ｔ個の集約演算を実行するステップと、

を含む請求項８記載の集約演算実行方法。

【請求項１０】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムにおいて、前記グループ毎に集約を行う演算を含む、Ｎ個の集約演算を前記データベースに対し実行する方法であって、

前記プロセッサの数と前記データベースの大きさと前記ネットワークの通信速度とを含む前記コンピュータ・システムのパラメータ及び実行する集約演算の数と各集約演算の結果を格納するメモリの量とを含む集約演算の性質に関するパラメータから、（ａ）各前記プロセッサが、前記自己のメモリ領域に、前記Ｎ個の集約演算のうちＭ個（Ｎ以下の整数）分の演算結果を記憶するスペースを確保するステップと、（ｂ）各前記プロセッサが、Ｍ個の集約演算をまとめて、前記データベースの自己の部分に対して実行するステップと、（ｃ）各前記プロセッサが、前記Ｍ個の集約演算の各プロセッサにおける演算結果を、集計すべきプロセッサに送信し、前記集計すべきプロセッサが最終結果を計算するステップと、

（ｄ）前記Ｎ個の集約演算の実行が終了するまで、前記ステップ（ａ）乃至（ｃ）を繰り返すステップとを含む第１集約演算実行方法と、（ｅ）各前記プロセッサが、前記Ｎ個の集約演算のうち自らが実行するＱ個の集約演算の演算結果を記憶するスペースを、前記自己のメモリ領域に確保するステップと、（ｆ）各前記プロセッサが、前記自己のメモリ領域に前記データベースの自己の部分である部分データベースの一部を読み出し、読み出された前記部分データベースの一部を前記ネットワークを介してブロードキャストすることを繰り返して、各前記プロセッサが全ての前記データベースのデータに対し前記自らが実行するＱ個の集約演算を実行するステップと、（ｇ）前記Ｎ個の集約演算が実行されるまで、前記ステップ（ｅ）及び（ｆ）を繰り返すステップとを含む第２集約演算実行方法と、（ｈ）前記Ｎ個の集約演算のうち実行するＴ個の集約演算を決定するステップと、

（ｉ）前記Ｔ個の集約演算を実行する際に集約される、各集約演算の各グループを取り扱うプロセッサを決定するステップと、（ｊ）各前記プロセッサが、前記データベースの自己の部分である部分データベースの一部を前記自己のメモリ領域に読み出し、読み出されたデータのうち他のプロセッサが集約すべき集約演算のグループに関するデータを当該集約演算のＩＤと共に前記他のプロセッサに前記ネットワークを介して送信し、自己が集約すべき集約演算のグループに関するデータについて前記Ｔ個の集約演算を実行するステップと、（ｋ）各前記プロセッサが集約すべき集約演算のグループに関する全てのデータに前記Ｔ個の集約演算を実行するまで、前記ステップ（ｊ）を実行するステップと、（ｌ）前記Ｎ個の集約演算を実行するまで、前記ステップ（ｈ）乃至（ｋ）を繰り返すステップとを含む第３集約演算実行方法とを含む複数の集約演算実行方法のうちいずれの方法が最も高速に実行できるかを決定するステップと、

決定された方法にて、前記Ｎ個の集約演算を実行するステップとを含む集約演算実行方法。

【請求項１１】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムであって、

前記グループ毎に集約を行う演算を含む、Ｎ個の集約演算を前記データベースに対し実行するために、前記複数のプロセッサの各々は、（ａ）前記自己のメモリ領域に、前記Ｎ個の集約演算のうちＭ個（Ｎ以下の整数）分の演算結果を記憶するスペースを確保するメモリ処理装置と、（ｂ）Ｍ個の集約演算をまとめて、前記データベースの自己の部分に対して実行するデータベース処理装置と、（ｃ）前記Ｍ個の集約演算の演算結果を、集計すべきプロセッサに送信する送信機と、

を有し、前記Ｎ個の集約演算の実行が終了するまで処理を繰り返し、前記集計すべきプロセッサは自己が集計すべき集約演算の演算結果を他のプロセッサから受信し、集計することを特徴とするコンピュータ・システム。

【請求項１２】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムであって、

前記グループ毎に集約を行う演算を含む、Ｐ個の集約演

算を前記データベースに対し実行するために、前記複数のプロセッサの各々は、（ａ）前記Ｐ個の集約演算のうち自らが実行するＱ個の集約演算の演算結果を記憶するスペースを、前記自己のメモリ領域に確保するメモリ処理装置と、（ｂ）前記自己のメモリ領域のワークスペースに前記データベースの自己の部分である部分データベースの一部を読み出し、読み出された前記部分データベースの一部を前記ネットワークを介してブロードキャストすることを繰り返して、各前記プロセッサが全ての前記データベースのデータに対し前記自らが実行するＱ個の集約演算を実行するデータベース処理装置と、
を有し、前記Ｐ個の集約演算が実行されるまで処理を繰り返すことを特徴とするコンピュータ・システム。
【請求項１３】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムであって、
前記グループ毎に集約を行う演算を含む、Ｓ個の集約演算を前記データベースに対し実行するために、
前記Ｓ個の集約演算のうち実行するＴ個の集約演算を決定し、当該Ｔ個の集約演算を実行する際に集約される、各集約演算の各グループを取り扱うプロセッサを決定するコントローラを含み、
各前記プロセッサは、
前記データベースの自己の部分である部分データベースの一部を読み出し、読み出されたデータのうち他のプロセッサが集約すべき集約演算のグループに関するデータを当該集約演算のＩＤと共に前記他のプロセッサに前記ネットワークを介して送信し、自己が集約すべき集約演算のグループに関するデータについて前記Ｔ個の集約演算を実行するデータベース処理装置を有し、
各前記プロセッサが集約すべき集約演算のグループに関する全てのデータに前記Ｔ個の集約演算を実行するまで前記データベース処理装置を動作させ、
前記Ｓ個の集約演算を実行するまで、前記コントローラ及び各前記プロセッサを動作させることを特徴とするコンピュータ・システム。
【請求項１４】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムであって、
前記グループ毎に集約を行う演算を含む、Ｎ個の集約演算を前記データベースに対し実行するために、
前記プロセッサの数と前記データベースの大きさと前記ネットワークの通信速度とを含む前記コンピュータ・システムのパラメータ及び実行する集約演算の数と各集約演算の結果を格納するメモリの量とを含む集約演算の性質に関するパラメータから、（ａ）各前記プロセッサが、前記自己のメモリ領域に、前記Ｎ個の集約演算のうちＭ個（Ｎ以下の整数）分の演算結果を記憶するスペースを確保するステップと、（ｂ）各前記プロセッサが、Ｍ個の集約演算をまとめて、前記データベースの自己の部分に対して実行するステップと、（ｃ）各前記プロセッサが、前記Ｍ個の集約演算の各プロセッサにおける演算結果を、集計すべきプロセッサに送信し、前記集計すべきプロセッサが最終結果を計算するステップと、
（ｄ）前記Ｎ個の集約演算の実行が終了するまで、前記ステップ（ａ）乃至（ｃ）を繰り返すステップとを含む第１集約演算実行方法と、（ｅ）各前記プロセッサが、前記Ｎ個の集約演算のうち自らが実行するＱ個の集約演算の演算結果を記憶するスペースを、前記自己のメモリ領域に確保するステップと、（ｆ）各前記プロセッサが、前記自己のメモリ領域に前記データベースの自己の部分である部分データベースの一部を読み出し、読み出された前記部分データベースの一部を前記ネットワークを介してブロードキャストすることを繰り返して、各前記プロセッサが全ての前記データベースのデータに対し前記自らが実行するＱ個の集約演算を実行するステップと、（ｇ）前記Ｎ個の集約演算が実行されるまで、前記ステップ（ｅ）及び（ｆ）を繰り返すステップとを含む第２集約演算実行方法と、（ｈ）前記Ｎ個の集約演算のうち実行するＴ個の集約演算を決定するステップと、
（ｉ）前記Ｔ個の集約演算を実行する際に集約される、各集約演算の各グループを取り扱うプロセッサを決定するステップと、（ｊ）各前記プロセッサが、前記データベースの自己の部分である部分データベースの一部を前記自己のメモリ領域に読み出し、読み出されたデータのうち他のプロセッサが集約すべき集約演算のグループに関するデータを当該集約演算のＩＤと共に前記他のプロセッサに前記ネットワークを介して送信し、自己が集約すべき集約演算のグループに関するデータについて前記Ｔ個の集約演算を実行するステップと、（ｋ）各前記プロセッサが集約すべき集約演算のグループに関する全てのデータに前記Ｔ個の集約演算を実行するまで、前記ステップ（ｊ）を実行するステップと、（ｌ）前記Ｎ個の集約演算を実行するまで、前記ステップ（ｈ）乃至（ｋ）を繰り返すステップとを含む第３集約演算実行方法とを含む複数の集約演算実行方法のうちいずれの方法が最も高速に実行できるかを決定するセレクタと、
決定された方法にて、前記Ｎ個の集約演算を実行するよう命ずる手段とを有するコンピュータ・システム。
【請求項１５】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムに、前記グループ毎に集約を行う演算を含む、Ｎ個の集約演算を前記データベースに対し実行させるプログラムを格納した記憶媒体であって、

前記プログラムは、（ａ）各前記プロセッサが、前記自己のメモリ領域に、前記Ｎ個の集約演算のうちＭ個（Ｎ以下の整数）分の演算結果を記憶するスペースを確保するステップと、（ｂ）各前記プロセッサが、Ｍ個の集約演算を同時に、前記データベースの自己の部分に対して実行するステップと、（ｃ）各前記プロセッサが、前記Ｍ個の集約演算の各プロセッサにおける演算結果を、集計すべきプロセッサに送信し、前記集計すべきプロセッサが最終結果を計算するステップと、（ｄ）前記Ｎ個の集約演算が終了するまで、前記ステップ（ａ）乃至（ｃ）を繰り返すステップとを前記コンピュータ・システムに実行させる、記憶媒体。

【請求項１６】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムに、前記グループ毎に集約を行う演算を含む、Ｐ個の集約演算を前記データベースに対し実行させるプログラムを記憶した記憶媒体であって、

前記プログラムは、（ａ）各前記プロセッサが、前記Ｐ個の集約演算のうち自らが実行するＱ個の集約演算の演算結果を記憶するスペースを、前記自己のメモリ領域に確保するステップと、（ｂ）各前記プロセッサが、前記自己のメモリ領域のワークスペースに前記データベースの自己の部分である部分データベースの一部を読み出し、読み出された前記部分データベースの一部を前記ネットワークを介してブロードキャストすることを繰り返して、各前記プロセッサが全ての前記データベースのデータに対し前記自らが実行するＱ個の集約演算を実行するステップと、（ｃ）前記Ｐ個の集約演算が実行されるまで、前記ステップ（ａ）及び（ｂ）を繰り返すステップとを前記コンピュータ・システムに実行させる、記憶媒体。

【請求項１７】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムに、前記グループ毎に集約を行う演算を含む、Ｓ個の集約演算を前記データベースに対し実行させるプログラムを格納した記憶媒体であって、

前記プログラムは、（ａ）前記Ｓ個の集約演算のうち実行するＴ個の集約演算を決定するステップと、（ｂ）前記Ｔ個の集約演算を実行する際に集約される、各集約演算の各グループを取り扱うプロセッサを決定するステップと、（ｃ）各前記プロセッサが、前記データベースの自己の部分である部分データベースの一部を前記自己のメモリ領域に読み出し、読み出されたデータのうち他のプロセッサが集約すべき集約演算のグループに関するデータを当該集約演算のＩＤと共に前記他のプロセッサに前記ネットワークを介して送信し、自己が集約すべき集約演算のグループに関するデータについて前記Ｔ個の集約演算を実行するステップと、（ｄ）各前記プロセッサが集約すべき集約演算のグループに関する全てのデータに前記Ｔ個の集約演算を実行するまで、前記ステップ（ｃ）を実行するステップと、（ｅ）前記Ｓ個の集約演算を実行するまで、前記ステップ（ａ）乃至（ｄ）を繰り返すステップとを前記コンピュータ・システムに実行させる、記憶媒体。

【請求項１８】ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されたコンピュータ・システムに、前記グループ毎に集約を行う演算を含む、Ｎ個の集約演算を前記データベースに対し実行させるプログラムを格納した記憶媒体であって、

前記プログラムは、

前記プロセッサの数と前記データベースの大きさと前記ネットワークの通信速度とを含む前記コンピュータ・システムのパラメータ及び実行する集約演算の数と各集約演算の結果を格納するメモリの量とを含む集約演算の性質に関するパラメータから、（ａ）各前記プロセッサが、前記自己のメモリ領域に、前記Ｎ個の集約演算のうちＭ個（Ｎ以下の整数）分の演算結果を記憶するスペースを確保するステップと、（ｂ）各前記プロセッサが、Ｍ個の集約演算をまとめて、前記データベースの自己の部分に対して実行するステップと、（ｃ）各前記プロセッサが、前記Ｍ個の集約演算の各プロセッサにおける演算結果を、集計すべきプロセッサに送信し、前記集計すべきプロセッサが最終結果を計算するステップと、

（ｄ）前記Ｎ個の集約演算の実行が終了するまで、前記ステップ（ａ）乃至（ｃ）を繰り返すステップとを含む第１集約演算実行方法と、（ｅ）各前記プロセッサが、前記Ｎ個の集約演算のうち自らが実行するＱ個の集約演算の演算結果を記憶するスペースを、前記自己のメモリ領域に確保するステップと、（ｆ）各前記プロセッサが、前記自己のメモリ領域に前記データベースの自己の部分である部分データベースの一部を読み出し、読み出された前記部分データベースの一部を前記ネットワークを介してブロードキャストすることを繰り返して、各前記プロセッサが全ての前記データベースのデータに対し前記自らが実行するＱ個の集約演算を実行するステップと、（ｇ）前記Ｎ個の集約演算が実行されるまで、前記ステップ（ｅ）及び（ｆ）を繰り返すステップとを含む第２集約演算実行方法と、（ｈ）前記Ｎ個の集約演算のうち実行するＴ個の集約演算を決定するステップと、

（ｉ）前記Ｔ個の集約演算を実行する際に集約される、各集約演算の各グループを取り扱うプロセッサを決定するステップと、（ｊ）各前記プロセッサが、前記データベースの自己の部分である部分データベースの一部を前記自己のメモリ領域に読み出し、読み出されたデータの

うち他のプロセッサが集約すべき集約演算のグループに関するデータを当該集約演算のＩＤと共に前記他のプロセッサに前記ネットワークを介して送信し、自己が集約すべき集約演算のグループに関するデータについて前記Ｔ個の集約演算を実行するステップと、（ｋ）各前記プロセッサが集約すべき集約演算のグループに関する全てのデータに前記Ｔ個の集約演算を実行するまで、前記ステップ（ｊ）を実行するステップと、（ｌ）前記Ｎ個の集約演算を実行するまで、前記ステップ（ｈ）乃至（ｋ）を繰り返すステップとを含む第３集約演算実行方法とを含む複数の集約演算実行方法のうちいずれの方法が最も高速に実行できるかを決定するステップと、
決定された方法にて、前記Ｎ個の集約演算を実行するステップとを含む記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、データ・マイニング（Data Mining）のようなデータベースの処理等に必要な集約演算（アグリゲーション：aggregation）の並列計算機上における処理方法に関する。なお、集約演算にはデータベース内のグループ毎に集約を行う演算を含む。

【０００２】

【従来の技術】データ・マイニングのような分野にてよく用いられる集約演算とは、関係データベースへの問い合わせ言語ＳＱＬのgroup-byのように、あるリレーションに対し、特定の属性が同一の値又同一の値の組若しくは所定の範囲に属する値を有するようなレコードを１つのグループとして、グループ毎に他の属性の合計、最大、最小、平均等を計算する処理である。このような集約演算は、最近、ＯＬＡＰ（On Line Analytical Processing）【詳しくは、E.F.Codd,S.B.Codd, and C.T.Salley, "Beyond decision support," Comupterworld, 27(30), July 1993参照のこと】やデータ・マイニング等の決定支援システムにおいて不可欠な操作となっている。

【０００３】以下集約演算の例を示す。表１は、データベースに記憶されている関係を表している。プロダクト番号、カスタマー番号、そして売上げについての列があり、各行は売上げが生ずるごとに記録されるようになっている。このような関係について、プロダクト番号ごとに売上げを合計するという演算を考える。この演算は、プロダクト番号ごとにグループが構成され、その各グループごとに合計を計算するという処理であり、上記集約演算の１つの例である。この集約演算を実施すると、表２のような結果を得ることができる。

【表１】

product#	customer#	sold
Ｇ１	Ｃ１	８
Ｇ１	Ｃ２	１０
Ｇ２	Ｃ２	５
Ｇ２	Ｃ３	１０
Ｇ３	Ｃ４	２

【表２】

product#	sold
Ｇ１	８
Ｇ２	２５
Ｇ３	２

【０００４】以前より、１つの集約演算を並列処理するためのアルゴリズムについては研究されてきた。以下、それらのアルゴリズムについて説明する。但し、各アルゴリズムにおいては、データベース中の全関係は各プロセッサに均等分割されていることを前提としている。

【０００５】１．２Ｐ　アルゴリズム

二段階にて実施されるため２Ｐという。

（１）第１フェーズとして、各プロセッサ（ノードともいう）は、そのプロセッサ用のディスク装置（データベースを記憶している）に対して、集約演算を実施する。

（２）第２フェーズとして、各プロセッサの結果は、集計用のプロセッサに集められ、最終結果を計算する。

【０００６】このような方法は、特開平５－２６１０号公報にも記載されている。この公報では、たかだか１つのグループについての集約演算しか取り扱っておらず、先に述べたようなデータ・マイニングのように複数のグループについて集約演算を実施することに何等の考慮もなされていない。

【０００７】２．Ｒｅｐ　アルゴリズム

リパーティション（repartition）アルゴリズムは以下のようなアルゴリズムにて実施される（図１）。

（１）最初に、各ノードが集約演算を実行すべきグループを決定する（ステップ１１０）。表１の例では、ノード１がプロダクト番号Ｇ１を担当し、ノード２がプロダクト番号Ｇ２を担当する、といったような割当てを決定する。

（２）次に、各ノードで、そのノード用のディスク装置（データベースを記憶した物）から、データの一部を読み出し、他のノードが集約すべきグループのデータならば、担当ノードに送信する（ステップ１２０）。例えば表１の例で、この表１がノード１用のディスク装置内に存在する場合、ノード１が表１の第２行目を読み出した時、ノード１は第２行目のデータをノード２に送信する。

（３）そして、各ノードで、他のノードから送られてきたデータを含む、自己が集約演算をすべきグループに関するデータに対して集約を実行する（ステップ１３０）。

【０００８】このような２つのアルゴリズムのうちどちらを用いると高速に処理できるかは、条件によって異なる。各アルゴリズムを用いて、１つの集約演算を処理した場合の所要時間の見積りを図２に示す。図２は、１６ノードのＩＢＭ　ＳＰ２（インタナショナル・ビジネス・マシーンズ・コーポレイションの商標）を用いた場合の見積りで、グループ数と応答時間の関係を示している。このシステムの場合、グループ数が2×10^5より少ない場合には、２Ｐアルゴリズムの方が高速であるが、グループ数が多い場合には、Ｒｅｐアルゴリズムの方が高速である。これまで、１つの集約演算の並列処理には、２ＰアルゴリズムとＲｅｐアルゴリズムを動的に切り換えるようなアルゴリズムが提案されている（例えば、Ambuj Shatdal and Jeffrey F. Naughton, "Adaptive parallel aggregation algorithms," In Proceedings of the ACM SIGMOD Conference on Management of Data, pages 104-114, May 1995.を参照のこと）。

【０００９】また、上記以外の方法としては、データベース中の全レコードをブロードキャストする方法がある（Dine Bitton, Haran Boral, David J DeWitt, and W. Kevin Wilkinson, "Parallel algorithms for the execution of relational database operations," ACM Trans. on Database Systems, 8(3):324-353, Sep. 1983.を参照のこと）。しかし、このようなアルゴリズム（以下、ＢＣアルゴリズムと呼ぶ。）は、プロセッサ間を接続するネットワークが低速であった頃には非実用的な方法であった。

【００１０】このＢＣアルゴリズムをまとめておく。

３．ＢＣ　アルゴリズム

（１）全グループの中で、各ノードが集約演算を実施すべきグループを決定する。Ｒｅｐアルゴリズムと同様である。

（２）各ノードは、自己のディスク装置内の全データを他の全てのノードに、ブロードキャストする。

（３）各ノードは、ブロードキャストされたデータ（自己のディスク装置内のデータを含む）に対して、自己が集約すべきグループについての集約演算を実施する。

【００１１】これまでは、１つの集約演算を並列に実行するアルゴリズムを説明したが、１つのプロセッサで複数の集約演算を実行する場合には、演算の順番を調整したり、演算間の関連性を用いて全体の処理を高速化する方法（例えば、Sameet Agrawal, Rakesh Agrawal, Prasad M. Deshpande, Ashish Gupta, Jeffrey F. Naughton, Raghu Ramakrishnan, and Sunita Sarawagi, "On the computation of multidimensional aggregates," In Proceedings of the 22nd VLDB Conference, Sep. 1996.を参照のこと）等が提案されている。しかし、並列計算機を用いて複数個の演算を同時に処理する方法については、提案されていない。

【００１２】

【発明が解決しようとする課題】複数の集約演算を並列に実行する処理は、上記の方法を複数回繰り返せば高速に実行できる、というわけではない。以上のように本発明は、複数の集約演算を並列に高速実行する方法を提供することを目的とする。

【００１３】現在注目を集めている決定支援システムのコアとなる技術であるＯＬＡＰやデータマイニングでは、複数個の集約演算を行う必要がある。例えば、多次元属性を持つデータの分析に対して、複数個の集約演算を実行するＤａｔａ　Ｃｕｂｅ（例えば、Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh, "Data cube: A relational aggregation operator generalizing group-by, cross-by, and sub-by, and sub-totals," Technical report, Microsoft , Nov. 1995等を参照のこと）オペレータが提案されている。また、データ・マイニングのあるアプリケーションは、複数の集約演算の結果を用いて属性間の関係を自動的に発見し、グラフィカルな表示を行う（例えば、Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama, "Data Mining optimizedassociation rules for numeric attributes, " In Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems, pages 182-191, June 1996, Takeshi Fukuda, Yasuhiko Morimoto, ShinichiMorishita, and Takeshi Tokuyama, "Data mining using two-dimensional optimized association rules: Scheme, algorithms, and Visualization," In proceedings of the ACM SIGMOD conference on Management of Data, pages 13-23, June 1996.等を参照のこと）。これらの技術では、インタラクティブに操作できることが要求されており、応答時間が重要な要素となっている。応答時間を短縮するための１つの方法として、事前に集約演算を実行しておくことが必要となる。よって、複数の集約演算を並列に高速実行することにより、ＯＬＡＰやデータ・マイニングの実行を高速にすることも目的である。

【００１４】さらに、複数の集約演算を実行する方法を、ハードウエアの条件や複数の集約演算の性質等に基づき、切り換えることができるようにし、様々な条件の下、複数の集約演算を常に同じ方法で実行するよりも、より高速に実行することも目的とする。

【００１５】

【課題を解決するための手段】以上の目的を実現するために最も重要な技術的事項は、ディスク装置へのアクセスの回数及び時間である。今や、ディスク装置へのアクセスは、並列処理装置の各プロセッサ間の通信速度より遅い。よって、このディスク装置へのアクセスをいかに減らすかが、本発明の目的達成に大きな影響を及ぼす。以下、このディスク装置へのアクセスを減らす具体的方法について説明する。

【００１６】本発明の第１の集約演算実行方法は以下の

ようなステップを含む。すなわち、（ａ）各プロセッサが、自己のメモリ領域に、Ｎ個の集約演算のうちＭ個（Ｎ以下の整数）分の演算結果を記憶するスペースを確保するステップと、（ｂ）各プロセッサが、Ｍ個の集約演算をまとめて、データベースの自己の部分に対して実行するステップと、（ｃ）各プロセッサが、Ｍ個の集約演算の各プロセッサにおける演算結果を、集計すべきプロセッサに送信し、集計すべきプロセッサが最終結果を計算するステップと、（ｄ）Ｎ個の集約演算の実行が終了するまで、ステップ（ａ）乃至（ｃ）を繰り返すステップと、である。なお、本発明の集約演算実行方法を実施するのに適当なコンピュータ・システムは、ネットワークにより接続された複数のプロセッサが各々自己のメモリ領域及び１又は複数のグループに分けることができるデータを含むデータベースの自己の部分を使用できるよう構成されている。メモリについては、各ノードごとに分離されて用意されていても、ネットワークに接続されたメモリが、幾つかのメモリ領域に分けられており、そのメモリ領域を各プロセッサが用いる方式にしてもよい。ディスクについても同様である。

【００１７】この第１の集約演算実行方法は、従来技術の欄で述べた２Ｐアルゴリズムをベースに考えられた方法である。この方法では、先に示したＩＢＭ　ＳＰ２で、結果のグループの数が比較的少ない場合に計算速度が高速になる。

【００１８】上記のステップ（ｂ）は、(b1) 各プロセッサが、自己のメモリ領域のワークスペースに、データベースの自己の部分である部分データベースの一部を読み出すステップと、(b2) 各プロセッサが、自己のメモリ領域に格納された本ステップ以前の演算結果と読み出された部分データベースの一部とについて、Ｍ個の集約演算を実行するステップと、(b3) 各プロセッサが、部分データベースのすべてに対してＭ個の集約演算が実行されるまで、ステップ(b1)及び(b2)を繰り返すステップとを含むようにすることもできる。ステップ（b2）は、読み出された部分データベースの一部についてのみＭ個の集約演算を実行して、それからそのステップ以前の演算結果との集計を計算するようにすることもできるが、このような方法では遅くなる。

【００１９】上で述べたＭは、Ｎ以下の整数として、他の条件と関係なく選択することも可能であるが、集約演算の演算結果のために自己のメモリ領域に確保できるスペースから決定すると、繰り返しのステップ（ステップ（ｄ））が少なくて済むので、より高速である。

【００２０】本発明の第２の集約演算実行方法は、（ａ）各プロセッサが、Ｐ個の集約演算のうち自らが実行するＱ個の集約演算の演算結果を記憶するスペースを、自己のメモリ領域に確保するステップと、（ｂ）各プロセッサが、自己のメモリ領域のワークスペースにデータベースの自己の部分である部分データベースの一部を読み出し、読み出された部分データベースの一部をネットワークを介してブロードキャストすることを繰り返して、各プロセッサが全てのデータベースのデータに対し自らが実行するＱ個の集約演算を実行するステップと、（ｃ）Ｐ個の集約演算が実行されるまで、ステップ（ａ）及び（ｂ）を繰り返すステップとを含む。第２の集約演算実行方法のためのコンピュータ・システムは、第１の集約演算実行方法と同様である。

【００２１】この第２の集約演算実行方法は、ＢＣアルゴリズムをベースに考えられた方法である。この方法では、第１の集約演算実行方法より比較的グループの数が多い場合に、計算速度がより高速になる。

【００２２】この第２の集約演算実行方法のステップ（ａ）は、(a1) １つの集約演算の演算結果を格納できるようなスペースが１つのプロセッサの自己のメモリ領域に存在するか検査するステップと、(a2) スペースが存在する場合には、スペースを１つの集約演算の演算結果のために確保するステップと、(a3) スペースが存在しない場合には、他のプロセッサのメモリ領域に前記１つの集約演算の演算結果を格納できるようなスペースが存在するか検査するステップと、(a4) 他のプロセッサのメモリ領域にスペースが存在する場合には、当該他のプロセッサのメモリ領域のスペースを前記１つの集約演算の演算結果のために確保するステップとを含むようにすることもできる。この時、他のプロセッサのメモリ領域にスペースが存在しない場合には、後の繰り返し処理にて前記１つの集約演算を実行するようにすることもできる。どうしても、ある集約演算にスペースが存在しないような場合も生じ得るが、その場合には当該集約演算を幾つかに分けて実行するようなことも可能である。

【００２３】Ｑについては、先のＭと同様に、集約演算の演算結果のために自己のメモリ領域に確保できるスペースから決定する、ということも可能である。

【００２４】さらに第２の集約演算実行方法のステップ（ｂ）は、(b1) 各プロセッサが、自己のメモリ領域のワークスペースに、部分データベースの一部を読み出すステップと、(b2) 読み出された部分データベースの一部をネットワークを介してブロードキャストするステップと、(b3) 自己のメモリ領域に格納された本ステップ以前の演算結果と読み出された部分データベースの一部と他のプロセッサから送られてきたデータとについて、自らが実行するＱ個の集約演算を実行するステップと、(b4) 各プロセッサが全てのデータベースの内容に対し自らが実行するＱ個の集約演算を実行するまでステップ(b1) 乃至 (b3) を繰り返すステップとを含むようにすることも可能である。ステップ (b3) で、読み出された部分データベースの一部と他のプロセッサから送られてきたデータとについてＱ個の集約演算を実行し、後に当該ステップ以前の演算結果と集計をとってもよいが、これでは遅くなる。

【００２５】本発明の第３の集約演算実行方法は、（ａ）Ｓ個の集約演算のうち実行するＴ個の集約演算を決定するステップと、（ｂ）Ｔ個の集約演算を実行する際に集約される、各集約演算の各グループを取り扱うプロセッサを決定するステップと、（ｃ）各プロセッサが、データベースの自己の部分である部分データベースの一部を自己のメモリ領域に読み出し、読み出されたデータのうち他のプロセッサが集約すべき集約演算のグループに関するデータを当該集約演算のＩＤと共に他のプロセッサにネットワークを介して送信し、自己が集約すべき集約演算のグループに関するデータについてＴ個の集約演算を実行するステップと、（ｄ）各プロセッサが集約すべき集約演算のグループに関する全てのデータにＴ個の集約演算を実行するまで、ステップ（ｃ）を実行するステップと、（ｅ）Ｓ個の集約演算を実行するまで、ステップ（ａ）乃至（ｄ）を繰り返すステップとを含む。第３の集約演算実行方法のためのコンピュータ・システムも、第１の集約演算実行方法と同様である。

【００２６】この第３の集約演算実行方法は、Ｒｅｐアルゴリズムをベースに考えられた方法である。この方法では、第２の集約演算実行方法より比較的グループの数が多い場合に、計算速度がより高速になる。

【００２７】第３の集約演算実行方法におけるステップ（ｃ）は、（c1）各プロセッサが、データベースの自己の部分である部分データベースの一部を自己のメモリ領域のワークスペースに読み出すステップと、（c2）各プロセッサが、読み出されたデータの各部分を必要とするプロセッサを求め、当該部分を関連する集約演算のＩＤと共に必要とするプロセッサにネットワークを介して送信するステップと、（c3）各プロセッサが、読み出されたデータのうち自己が集約すべき集約演算のグループに関するデータ及び他のプロセッサからのデータ及び自己のメモリ領域に格納された本ステップ以前の演算結果に対し、Ｔ個の集約演算を実行するステップとを含むようにすることも考えられる。ネットワークを介しての送信は、Ｔ個の集約演算と同時に行ってもよい。

【００２８】この第３の集約演算実行方法におけるＴ個の集約演算の決定方法及びどのプロセッサが集約演算のどのグループを集約するかについては、各プロセッサの使用するメモリの容量によって決められる。

【００２９】ところで、第１乃至第３の集約演算実行方法のうち最適なアルゴリズムは、条件によって異なる。よって、プロセッサの数とデータベースの大きさとネットワークの通信速度とを含むコンピュータ・システムのパラメータ及び実行する集約演算の数と各集約演算の結果を格納するメモリの量とを含む集約演算の性質に関するパラメータを用いて、上記のうち最も高速なアルゴリズムを選び出し、最適なアルゴリズムによって複数の集約演算を実行するようにすれば、ユーザは常に高速に複数の集約演算の結果を得ることができる。なお、選択は３つの方法の中から行わなければならないわけではなく、他の方法を含めてその中で選択することも、３つの方法のサブセットの中から選択することも可能である。

【００３０】以上の述べた方法は、プログラムによって実現することも、専用の装置を用いて実現することも可能である。このように実現形態を変更することは、以下の説明を理解した当業者が通常行うことのできる事項である。

【００３１】

【発明の実施の形態】初めに本発明において用いられるコンピュータ・システムのハードウエア構成について説明する。図３は、その典型的な一例を示すものである。プロセッサ・エレメントＰＥ１（３ａ）、プロセッサ・エレメントＰＥ２（３ｂ）、及びプロセッサ・エレメントＰＥｎ（３ｃ）がネットワーク１を介して接続されている。このプロセッサ・エレメントは図３のように３つである必要はなく、複数であればよい。一般にｎ個とする。プロセッサ・エレメントＰＥ１（３ａ）には、メモリ５ａ及びディスク装置７ａが接続されており、同様にプロセッサ・エレメントＰＥ２（３ｂ）にはメモリ５ｂ及びディスク装置７ｂ、プロセッサ・エレメントＰＥｎ（３ｃ）にはメモリ５ｃ及びディスク装置７ｃが接続されている。各ディスク装置には、データベースの一部分がそれぞれ記憶されている。ネットワーク１は、プロセッサ・エレメント間の通信に用いられ、高速にデータをやり取りできるようになっている。

【００３２】図３は一例であって、例えば図４のように、各プロセッサ・エレメントに直接メモリが接続されておらず、メモリ９及びメモリ１０がネットワーク１に接続されるような構成も可能である。この場合メモリ９は、プロセッサ・エレメントＰＥ１用の領域９ａとプロセッサ・エレメントＰＥ２用の領域９ｂとに分けることができる。メモリ９はこのようにそれぞれのプロセッサ・エレメント用の領域の他、プロセッサ・エレメント間で共有する領域を設けてもよい。また、メモリ１０はＰＥｎ専用となっているが、このように、ネットワーク１に各プロセッサ・エレメント用に別のメモリを接続するような構成とすることも可能である。プロセッサ・エレメント３ａ，３ｂ，３ｃは、ディスク装置７ａ，７ｂ，７ｃからデータを読み出すと、ネットワーク１を介して自己のメモリ領域９ａ，９ｂ，１０のワークエリアにデータを書き込む。

【００３３】また図５のように、プロセッサ・エレメントとメモリは図３と同様な接続をしているが、ディスク装置がネットワーク１に接続されているような構成も考えられる。ここで、ディスク装置１１はプロセッサ・エレメントＰＥ１用のデータ１１ａとプロセッサ・エレメントＰＥ２用のデータ１１ｂとを有している。ディスク装置１１は、プロセッサ・エレメントＰＥ１及びＰＥ２用データの他、共有のデータ等を含むことができる。ま

た、ディスク装置１３は、プロセッサ・エレメントＰＥｎ用のデータを格納している。このように、各プロセッサ・エレメント専用のディスク装置をネットワーク１に別々接続する構成も可能である。これらディスク装置内のデータは、ネットワーク１を介してプロセッサ・エレメント３ａ，３ｂ，３ｃによって読み出され、メモリ５ａ，５ｂ，５ｃのワークエリアに取り込まれる。

【００３４】図３、図４、及び図５を幾つか組み合わせた構成も可能である。例えば、幾つかのプロセッサ・エレメントは図３のような構成をとっており、幾つかのプロセッサは図４や図５のような構成をすることも可能である。また、図３，図４，図５以外の構成も可能であり、要するに、プロセッサ間でメモリやディスク装置を共有する必要はない。

【００３５】以上のようなハードウエア構成を前提として、以下どのような処理を行うかを説明する。なお、以下に説明する処理では、システム全体のための処理も一部に含まれるが、その処理は複数のプロセッサ・エレメントのうち１つをその処理のために割当てたり、そのような処理専用のプロセッサを用意してもよい。

【００３６】１．２Ｐｍ　アルゴリズム（図６）

本アルゴリズムは、従来技術で説明した２Ｐアルゴリズムをベースとしたものであり、複数の集約演算を並列に実施するように拡張したものであるため、２Ｐｍアルゴリズムと呼ぶことにする。複数の集約演算を実施する際に、ディスク装置へのアクセスを最小限にすることで高速化する。なお、全部でＮ個の集約演算を実行することを前提とする。

【００３７】（１）最初に、自己のメモリ領域にＭ個の集約演算の演算結果を記憶するスペースを確保する（ステップ２１０）。このＭ個はその演算結果を各プロセッサ・エレメントのメモリ領域に格納できる範囲で決められるが、なるべくたくさんの集約演算を１度に実行する方がディスク・アクセスを共有することができるので、処理が高速になる。また、本アルゴリズムを実行する全てのプロセッサ・エレメントが、全て同じＭ個の集約演算を実行する方が高速処理できる。これは、後に説明する集計処理が必要となるため、プロセッサ・エレメント毎に異なる個数の集約演算を実施していると、集計処理ができない演算結果の一部がメモリに残ってしまい、他のプロセッサ・エレメントで演算の実行が終了するのを待たなければならないといった問題が生じるからである。よって、実行する集約演算は、本アルゴリズムを実行するプロセッサ・エレメントが同じ個数の同じ集約演算を１度に実行できるという観点から選択すると、処理の高速化に効果的である。なお、演算結果を記憶するスペースを演算を実行する前に知らなければならないが、これは少しのデータに対して試算をしてみれば、演算結果の規模は把握することができ、大きな負担ではない。

【００３８】以下に、ステップ２１０の擬似コードを示しておく。

【表３】

```
// 変数
 N   : number of node
 mem : size of memory
 MEM : variable ( 0 ≦ MEM ≦ mem )
 q     : query
 NL    : list of queries
 tmpNL : list of queries # 次回以降に実行する分
 ML    : list of queries # 今回実行する分
// 初期化
 NL    = { q_1, q_2, ..., q_i,... ,q_Q }  # list of Q queries.
 mem = getMaxMemSize() # メモリサイズを得る。
                       # 各ノードが協調して最も小さいサイズに合わせる
 # 各ノードが以下を実行
 while NL ≠ NIL 且つ tmpNL ≠ NIL do     #  NIL : 空リスト
  begin
   MEM = mem ;
   tmpNL = NIL ;
   ML    = NIL ;
   while NL ≠ NIL do     # NL が 空になるまで実行
    begin

     q = pop( NL );       # pop() リストの先頭を取ってくる
     if  ( MEM >= |q| ) then  # q のメモリが確保できるか調べる
      begin                  # |q| は演算 q が使用するメモリサイズ
```

```
    append( ML, q ) ;     # ML に q を加える
    MEM = MEM - |q| ;     # 残りのメモリサイズから |q| を引く
   end
  else
   begin
    append( tmpML, q ) ;  # 演算 q を未処理リストに加える
   end
 end
 # ML 中の query について集約演算を実行
 # tmpML は、次回以降に
 while ML ≠ NIL do       # ML が NIL になるまで実行
  begin
   q = pop( ML ) ;       # MLから先頭の演算を取ってくる
   allocate ( q ) ;      # 演算 q 用のメモリを確保
  end
 2Pm( ) ;                # ML のリストにあった query について
                         # 2Pmアルゴリズムを実行
 copy( NL, tmpNL ) ;     # NL に、tmpNL をコピー
end
```

【００３９】（２）データベースの自己の部分に対してＭ個の集約演算を実行する（ステップ２２０）。この処理をより詳しく説明すると、最初に自己のメモリ領域のワークスペースに、データベースの自己の部分のうち、今回処理する分を読み出す。そして、自己のメモリ領域に格納された、これまでの演算結果（一番最初のステップにおいては存在しない）と、データベースの自己の部分のうち、読み出された今回処理する分とについて、Ｍ個の集約演算を実行する。この２つのステップを、データベースの自己の部分すべてに対して実行するまで繰り返す。具体例については後に説明する。１回読み出したデータに対し、従来の方法では１つの集約演算しか行われていなかったため、そのままＭ個の集約演算を実行する場合に適用すると、Ｍ回データを読み出さなければならなかった。一方、本発明では、１回読み出したデータに対し、Ｍ個の集約演算が実施されているため、ディスク・アクセスのための時間は全体として１／Ｍとなる。

【００４０】（３）そして、Ｍ個の集約演算の演算結果を集計担当プロセッサ・エレメント（集計担当ノードともいう）に送信し、集計担当プロセッサ・エレメントが最終結果を計算する（ステップ２３０）。集計担当プロセッサ・エレメントは、１つでも複数でもよい。例えば、Ｍ個のプロセッサ・エレメントが共に本アルゴリズムを実行している場合には、各プロセッサが１つの集約演算の演算結果の集計担当プロセッサ・エレメントとし、最終結果を計算するようにすることもできる。Ｍ個よりプロセッサ・エレメントが少ない場合には、１つのプロセッサ・エレメントが複数個の集約演算の演算結果を集計してもよいし、ただ１つのプロセッサ・エレメントがＭ個の集約演算の演算結果をすべて集計するようにしてもよい。どのプロセッサ・エレメントがどの集約演算の演算結果を集計するかは、予め決めておく。

【００４１】また、集約演算の演算結果における各グループをプロセッサ・エレメントごとに割当て、集計する方法も考えられる。例えば、プロセッサ・エレメント１が集約演算１のグループ１及び２を集計し、プロセッサ・エレメント２が集約演算１のグループ３及び４と集約演算２のグループ１及び２を集計する、といった具合である。

【００４２】（４）Ｍ個の集約演算の最終結果が得られると、Ｎ個の集約演算が全て実行されたか判断される（ステップ２４０）。もし、Ｎ個の集約演算がすべて実行されたわけではない場合には、ステップ２１０に戻って処理を繰り返す。この場合、通常はもとのＭと異なるＭで繰り返しを実行する。もし、Ｎ個の集約演算がすべて実行された場合には、処理を終了する。

【００４３】では実際の実行例を図７乃至図１０の例を用いて説明する。図７はプロセッサ・エレメント１用のディスク装置に記憶されたリレーションを、図８はプロセッサ・エレメント２用のディスク装置に記憶されたリレーションを、図９はプロセッサ・エレメント３用のディスク装置に記憶されたリレーションを、図１０はプロセッサ・エレメント４用のディスク装置に記憶されたリレーションを、それぞれ表し、コンピュータ・システムには４つのプロセッサ・エレメントが存在するとする。表の各列は、右から売上げの月、日、曜日、場所、店番号、製品番号、売上げ数をそれぞれ表す。表の各行は、売上げが発生した時に生成されるとする。

【００４４】月ごとに売上げ数を合計する演算（演算１）と、曜日ごとに売上げ数を合計する演算（演算２）

と、場所ごとに売上げ数を合計する演算（演算３）と、製品番号ごとに売上げ数を合計する演算（演算４）と、場所及び曜日ごとに売上げ数を合計する演算（演算５）とを一度に実行することとする。

【００４５】プロセッサ・エレメント１は、図７の第１行目のデータをディスク装置からメモリのワークスペースに読み出す。そして、演算１を実施するため、月と売上げ数の列を参照する。月は"Ａｐｒ"で売上げ数"３"が最初に演算１の結果として記憶される。次に、演算２を実行するため、曜日と売上げ数の列を参照し、曜日は"Ｓｕｎ"で売上げが"３"が演算２の結果として記憶される。演算３を実行するために、場所と売上げ数の列を参照し、場所"Ｋｙｕｕｓｈｕｕ"で売上げ数"３"が演算３の結果として記憶される。演算４を実行するために、製品番号と売上げ数の列を参照し、製品番号２で売上げ数"３"が演算４の結果として記憶される。演算５を実行するために、場所、曜日、売上げ数の列が参照され、場所及び曜日が"Ｋｙｕｕｓｈｕｕ""Ｓｕｎ"で売上げ"３"が演算５の結果として記憶される。

【００４６】次にプロセッサ・エレメント１は、図７の第２行目のデータをディスク装置からメモリのワークスペースに読み出す。そして、演算１を実行するため、月と売上げ数の列を参照する。ここで、読み出したデータの月のデータは、先に記憶した最初の演算１の結果の月とは異なるため、新たに月は"Ａｕｇ"で売上げ"４"を演算１の結果に追加する。次に、演算２を実行するため、曜日と売上げ数の列を参照する。ここで、読み出したデータの曜日のデータとこれまでの演算２の実行結果の曜日のデータとは同じなので、演算２の結果の曜日"Ｓｕｎ"の項目に売上げ"４"を足して"７"を記憶する。演算３を実行するため、場所と売上げ数の列を参照する。ここで、読み出したデータの場所のデータは、先に記憶した、これまでの演算３の結果の場所とは異なるため、新たに場所は"Ｈｏｋｋａｉｄｏ"で売上げ数"４"を演算３の結果に追加する。演算４を実行するため、製品番号と売上げ数の列を参照する。ここで読み出したデータの製品番号のデータは先に記憶した、これまでの演算４の結果の製品番号とは異なるので、新たに製品番号"１"で売上げ"４"を演算４の結果に追加する。演算５を実行するため、場所及び曜日と売上げ数の列を参照する。ここで読み出したデータの場所及び曜日は、先に記憶した、これまでの演算５の結果の場所及び曜日とは異なるため、新たに場所及び曜日が"Ｈｏｋｋａｉｄｏ""４"を演算５の結果に追加する。

【００４７】次にプロセッサ・エレメント１は、図７の第３行目のデータをディスク装置からメモリのワークスペースに読み出す。演算１を実行するため、月と売上げ数の列を参照する。ここで、読み出したデータの月のデータは、これまでの演算１の結果の月とは異なるため、新たに月は"Ｊｕｎ"で売上げ"２"を演算１の結果に追加する。演算２を実行するため、曜日と売上げ数の列を参照する。ここで、読み出したデータの曜日はこれまでの演算２の結果の曜日とは異なるため、新たに曜日は"Ｍｏｎ"で売上げ数"２"を追加する。演算３を実行するために、場所と売上げ数の列を参照する。読み出したデータの場所とこれまでの演算３の結果の場所とは一致するものがある。よって、場所"Ｈｏｋｋａｉｄｏ"で売上げ数"４"の項目に売上げ数"２"を足して"６"を記憶する。演算４を実行するために、製品番号と売上げ数の列を参照する。読み出したデータの製品番号のデータと、これまでの演算４の結果の製品番号とは一致するものがある。よって、製品番号"１"で売上げ数"４"の項目に売上げ数"２"を足して"６"を記憶する。演算５を実行するために、場所及び曜日と売上げ数の列を参照する。ここで、読み出したデータの場所及び曜日のデータは、これまでの演算５の結果の場所及び曜日とは異なるため、新たに場所及び曜日"Ｈｏｋｋａｉｄｏ""Ｍｏｎ"で売上げ数"２"を追加する。

【００４８】ここまでの処理で、プロセッサ・エレメント１のメモリには図１１の結果が記憶される。このような処理をプロセッサ・エレメント１のディスク装置に記憶された図７のリレーションに対し繰り返すと、プロセッサ・エレメント１のメモリには図１２のような結果が記憶されることになる。

【００４９】プロセッサ・エレメント２、プロセッサ・エレメント３、及びプロセッサ・エレメント４でも同様の処理が実施され、ディスク装置に記憶された図８、図９、及び図１０のリレーションに対する演算１乃至演算５の結果は、それぞれ図１３、図１４、及び図１５に表わされるようになる。

【００５０】そして、集計処理を実施すれば、集約演算１乃至５の最終結果を得ることができる。本例では、各プロセッサ・エレメントが集約演算１乃至５の一部のグループについて集計するという集計方法を採用している。すなわち、プロセッサ・エレメント１は、演算１の月"Ｊａｎ"と"Ｆｅｂ"のグループ、演算２の曜日"Ｍｏｎ"と"Ｔｕｅ"のグループ、演算３の場所"Ｈｏｋｋａｉｄｏ"のグループ、演算４の製品番号"１"のグループ、演算５の場所及び曜日のうち場所"Ｈｏｋｋａｉｄｏ"のグループについて集計する。プロセッサ・エレメント２は、演算１の月"Ｍａｒ"と"Ａｐｒ"のグループ、演算２の曜日"Ｗｅｄ"と"Ｔｈｕ"のグループ、演算３の場所"Ｋａｎｔｏ"のグループ、演算４の製品番号"２"のグループ、演算５の場所及び曜日のうち場所"ｋａｎｔｏ"のグループについて集計する。

【００５１】さらに、プロセッサ・エレメント３は、演算１の月"Ｍａｙ"と"Ｊｕｎ"のグループ、演算２の曜日"Ｆｒｉ"と"Ｓａｔ"のグループ、演算３の場所"Ｋａｎｓａｉ"のグループ、演算４の製品番号"１"のグループ、演算５の場所及び曜日のうち場所"Ｋａｎｓａｉ"の

グループについて集計する。プロセッサ・エレメント４は、演算１の月"Ｊｕｌ"と"Ａｕｇ"のグループ、演算２の曜日"Ｓｕｎ"のグループ、演算３の場所"Ｋｙｕｕｓｈｕｕ"のグループ、演算５の場所及び曜日のうち場所"Ｋｙｕｕｓｈｕｕ"のグループについて集計する。

【００５２】よって、各プロセッサ・エレメントは、自己が保持しているグループのデータのうち、自己が集計するグループのデータ以外のデータを集計担当ノードに送信する必要がある。

【００５３】以上、プロセッサ・エレメント１が集計した結果を図１６、プロセッサ・エレメント２が集計した結果を図１７、プロセッサ・エレメント３が集計した結果を図１８、プロセッサ・エレメント４が集計した結果を図１９に示す。

【００５４】２．ＢＣｍ　アルゴリズム（図２０）

本アルゴリズムは、従来技術で説明したＢＣアルゴリズムをベースとしたものであり、複数の集約演算を並列に実行するように拡張したものであるため、ＢＣｍアルゴリズムと呼ぶことにする。本アルゴリズムも、複数の集約演算を実行する際に、ディスク装置へのアクセスを最小限にすることで高速化する。なお、全部でＰ個の集約演算を実行することを前提とする。

【００５５】（１）まず、自らが実行するＱ個の集約演算の演算結果を記憶するスペースを自己のメモリ領域に確保する（ステップ３１０）。このＱ個は、２Ｐｍアルゴリズムで説明したＭ個と同様に、各プロセッサ・エレメントのメモリ領域に格納できる範囲で決められるが、なるべくたくさんの集約演算を一度に実行する方がディスク装置へのアクセスを共有することになるので、処理が高速になる。なお、Ｑ個の集約演算は、各プロセッサ・エレメントで異なる集約演算を実行するようにしてもよいし、各プロセッサ・エレメントで同一にしておき、集約演算の結果のグループを分けるということも可能である。

【００５６】どの集約演算をどのプロセッサ・エレメントに割当てるかという処理の一例を図２１に示す。最初に未実行の集約演算を１つ選択する（ステップ４１０）。そして、プロセッサ・エレメントを１つ選択する（ステップ４２０）。その後、選択されたプロセッサ・エレメントのメモリに選択された集約演算の結果を記憶するスペースがあるか検査する（ステップ４３０）。もし、そのプロセッサ・エレメントのメモリにそのスペースがあるならば、そのプロセッサ・エレメントに選択された集約演算を割当てる（ステップ４４０）。一方、選択されたプロセッサ・エレメントのメモリにその集約演算の演算結果を格納するスペースがない場合には、他のプロセッサ・エレメントのメモリについて検査する（ステップ４５０）。但し、ある集約演算についてすべてのプロセッサ・エレメントについて検査したにもかかわらず、その演算結果用のスペースが確保できなかった場合には（ステップ４６０）、当該集約演算は別途処理する（ステップ４７０）。別途処理というのは、図２０のステップ３５０で繰り返し処理を実行するようになっているので、後の繰り返し処理の時に実行するようにするか、場合によっては、当該集約演算を幾つかのグループに分割して実行するように変更する等の処理を含む。

【００５７】そして、他の集約演算についてプロセッサ・エレメントの割当てを決定するわけであるが、全てのプロセッサ・エレメントのメモリに既に十分なスペースがない場合は図２０のステップ３５０による繰り返し処理の際に再度割当てを行うので、又、全ての未実行の集約演算について割当て処理を実行してしまった場合（ステップ４８０）には、割当ての済んだ集約演算を実行するため、この処理は一旦終了する（ステップ４９０）。

【００５８】集約演算のプロセッサ・エレメントへの上記の割当て処理は一例であって、他の方法によって割当てすることもできる。ユーザが、意図的にある集約演算を特定のプロセッサ・エレメントに割当てることができるようにすることも可能である。なお、先にも述べたが、各プロセッサ・エレメントで同一集約演算を実行するが、各プロセッサ・エレメントで集約するグループを変えることもできる。この場合には、同一種類の集約演算であるがグループが異なるため別集約演算として図２１のプロセスにてプロセッサ・エレメントを割当てることもできる。また、このような実行方法のために、別の割当てアルゴリズムを用意することも可能である。

【００５９】図２１をより詳しくした擬似コードを以下に示す。

【表４】

```
// 変数
 N   : number of node
 m_i : memory size for each node ( 1 ≦ i ≦ N )
 q   : query
 q_i : query ( 1 ≦ i ≦ Q )
 QL    : list of queires to be aggregated # 集約すべき演算のリスト
 tmpQL : list of queries  # 次回以降実行する分
 ML    : list of queries  # 今回実行する分
// 初期化
 QL    = { q_1, q_2, ...., q_i.... ,q_Q }  # list of Q queries.
```

```
while QL ≠ NIL 且つ tmpQL ≠ NIL do    # NIL : 空リスト
 begin
  for i=0 to N do
   begin
    m_j = getMaxMemSize( j ) ; # 各ノードのメモリサイズを得る。
   end
  tmpQL = NIL ;
  ML    = NIL ;
  while QL ≠ NIL do     # QL が 空になるまで実行
   begin

    q = pop( QL );       # pop() リストの先頭を取ってくる
    for j=1 to N do
     begin

      if ( m_j >= |q| ) then  # q のメモリが確保できるか調べる
       begin                  # |q| は演算 q が使用するメモリサイズ

        append( ML, q ) ;     # ML に q を加える
        m_j = m_j - |q| ;     # 残りのメモリサイズから |q| を引く
        break ;             # for 文を抜ける
       end
     end
     if (q の割り当て先が見つからない) then
     begin
      append( tmpQL, q ) ; # 演算 q を未処理リストに加える
     end
   end
   # ML 中の query について集約演算を実行
   # tmpQL は、次回以降に
   while ML ≠ NIL do      # ML が NIL になるまで実行
    begin
     q = pop( ML ) ;      # MLから先頭の演算を取ってくる
     allocate ( q ) ;     # 演算 q 用のメモリを確保
    end
   BQm( ) :              # ML のリストにあった query について
                         # BQmアルゴリズムを実行
   copy( QL, tmpQL ) :   # QL に、tmpQL をコピー
 end
```

【００６０】（２）図２０に戻って、データベースの自己の部分のうち、今回処理する分を読み出し、読み出したデータをネットワークを介してブロードキャストする（ステップ３２０）。ブロードキャストであるから、同じ内容のデータが全てのプロセッサ・エレメントに送信される。この際、どのデータがどのプロセッサ・エレメントで必要となるかは判断しなくともよい。先に述べたＲｅｐアルゴリズム及び後に述べるＲｅｐｍアルゴリズムにおいては、どのプロセッサがどのグループについて集約を行っているか又はどのプロセッサがどの集約演算のどのグループについて集約を行っているか判断して、必要なプロセッサ・エレメントに送信するようになっているが、本アルゴリズムにおいては、全てのプロセッサ・エレメントが同一集約演算を実行しているわけではないからである。全プロセッサ・エレメントで用いないようなデータが存在することが予め分かっている場合には、そのデータの送信を省略できる。

【００６１】（３）そして、他のプロセッサ・エレメントからのデータと、自ら読み出したデータと、これまでの演算結果（一番最初の場合には存在しない）に対し

て、Ｑ個の集約演算を実行する（ステップ３３０）。
（４）この（２）（ステップ３２０）（３）（ステップ３３０）の処理を、全てのデータベースのデータについてＱ個の集約演算を実行し終わるまで繰り返す（ステップ３４０）。個々のプロセッサ・エレメント用のデータベースの大きさにばらつきがある場合には、自己のデータベースを全て処理しても他のプロセッサ・エレメントからデータが送信されてきたり、逆に他のプロセッサ・エレメントからは送られてこないが自己のデータベースを全て処理していないという場合も考えられる。この場合、ステップ３３０は、自己のデータベース又は他プロセッサ・エレメントからの未処理のデータとそれまでの演算結果とに対しＱ個の集約演算を実行する、ということになる。
【００６２】（５）全てのデータベースのデータに対してＱ個の集約演算を実行した場合には、Ｐ個の集約演算すべてについて実行が終了したか判断する（ステップ３５０）。終了していない場合には、別の集約演算に対してステップ３１０以下を実行する。終了している場合には、全処理を終了する（ステップ３６０）。
【００６３】このＢＣｍアルゴリズムでは、ある集約演算を実行するプロセッサ・エレメントが、全てのデータについて演算を実行するが、ディスク装置へのアクセスは、自己のディスク装置に対してのみなので、このディスク装置へのアクセスのコストは削減されている。また、複数個の集約演算でこのディスク装置へのアクセスを共有するので、全体としてコストは削減されている。
【００６４】以上の処理を図７，図８，図９，図１０を用いて具体的に説明する。ここでは、プロセッサ・エレメント１が先に示した演算１を、プロセッサ・エレメント２が演算２及び３を、プロセッサ・エレメント３が演算４を、プロセッサ・エレメント４が演算５を実行するものとする。プロセッサ・エレメント２の動作を説明する。なお、１行ずつ読み出すような説明であるが、任意の行数ずつ読み出すようにすることも可能である。
【００６５】プロセッサ・エレメント２は、図８の第１行目をメモリのワークスペースに読み出す。そして、ネットワークを介して全プロセッサ・エレメントに送信する。他のプロセッサ・エレメントも同様に読み出したデータ（各データベースの第１行目）を全プロセッサ・エレメントに送信するので、プロセッサ・エレメント２のメモリのワークスペースには、図７，図８，図９及び図１０の第１行目が格納されている。そして、先に示した演算２及び演算３を実行する。すなわち、曜日ごとに売上げ数を合計する演算（演算２）と、場所ごとに売上げ数を合計する演算（演算３）である。
【００６６】プロセッサ・エレメント２がワークスペースに存在するデータに対し演算２を実行すると、曜日"Ｓａｔ"で売上げ数"７"、曜日"Ｗｅｄ"で売上げ数"１５"、曜日"Ｓｕｎ"で売上げ数"３"が得られる。また、演算３を実行すると、場所"Ｋｙｕｕｓｈｕｕ"で売上げ数"１０"、場所"Ｋａｎｓａｉ"で売上げ数"６"、場所"Ｈｏｋｋａｉｄｏ"で売上げ数"９"が得られる。
【００６７】この後、プロセッサ・エレメント２は、図８の第２行目をメモリのワークスペースに読み出す。そして、ネットワークを介して全プロセッサ・エレメントに送信する。他のプロセッサ・エレメントも同様に読み出したデータ（各データベースの第２行目）を全プロセッサ・エレメントに送信するので、プロセッサ・エレメント２のメモリのワークスペースには、図７，図８，図９及び図１０の第２行目が格納されるている。
【００６８】プロセッサ・エレメント２がワークスペースのデータ及びこれまでの演算結果に対して演算２を実行すると、曜日"Ｓａｔ"で売上げ数"１６"、曜日"Ｗｅｄ"で売上げ数"１５"、曜日"Ｓｕｎ"で売上げ数"７"、曜日"Ｔｕｅ"で売上げ数"８"、曜日"Ｔｈｕ"で売上げ数"１"が得られる。また、同様に演算３を実行すると、場所"Ｋｙｕｕｓｈｕｕ"で売上げ数"１１"、場所"Ｋａｎｓａｉ"で売上げ数"６"、場所"Ｈｏｋｋａｉｄｏ"で売上げ数"２１"、場所"Ｋａｎｔｏ"で売上げ数"９"が得られる。
【００６９】さらに、プロセッサ・エレメント２は、図８の第３行目をメモリのワークスペースに読み出す。そして、ネットワークを介して全プロセッサ・エレメントに送信する。他のプロセッサ・エレメントも同様に読み出したデータ（各データベースの第３行目）を全プロセッサ・エレメントに送信するので、プロセッサ・エレメント２のメモリのワークスペースには、図７，図８，図９及び図１０の第３行目が格納されるている。
【００７０】プロセッサ・エレメント２がワークスペースのデータ及びこれまでの演算結果に対して演算２を実行すると、曜日"Ｓａｔ"で売上げ数"１６"、曜日"Ｗｅｄ"で売上げ数"１５"、曜日"Ｓｕｎ"で売上げ数"１５"、曜日"Ｔｕｅ"で売上げ数"８"、曜日"Ｔｈｕ"で売上げ数"９"、曜日"Ｍｏｎ"で売上げ数"４"が得られる。また、同様に演算３を実行すると、場所"Ｋｙｕｕｓｈｕｕ"で売上げ数"１１"、場所"Ｋａｎｓａｉ"で売上げ数"６"、場所"Ｈｏｋｋａｉｄｏ"で売上げ数"３１"、場所"Ｋａｎｔｏ"で売上げ数"１９"が得られる。
【００７１】以上のような処理を繰り返し、図７，図８，図９及び図１０の全てのデータに対して、プロセッサ・エレメント２が演算２及び演算３を実行すると、図２２のようなデータが、プロセッサ・エレメント２のメモリに記憶されることになる。なお、プロセッサ・エレメント１、プロセッサ・エレメント３、プロセッサ・エレメント４の最終実行結果を図２３に示しておく。
【００７２】３．Ｒｅｐｍアルゴリズム（図２４）
本アルゴリズムは、従来技術の欄で説明したＲｅｐアルゴリズムをベースとしたものであり、複数の集約演算を並列に実施するように拡張したものであるため、Ｒｅｐ

mアルゴリズムと呼ぶことにする。複数の集約演算を実施する際に、ディスク装置へのアクセスを最小限にすることで全体を高速化する。なお、全部でＳ個の集約演算を実行するものとする。

【００７３】（１）最初に、Ｓ個の集約演算のうちどの集約演算を実行するか決定する（ステップ５１０）。ここで実行される集約演算の個数をＴ個とする。以下の説明では、同時に実行する集約演算は、各プロセッサについて同一にし、その代わりにグループの種類を変えるようにする。但し、必ず各プロセッサで実行する集約演算が同一である必要はない。また、このステップは、システム全体で使用可能なメモリ容量を勘案して実施されるべきであるが、必ずしもメモリ容量のみにより制限する必要はなく、ユーザがメモリ容量の限度において指定するようにしてもよい。以下に、本ステップの実施方法の一例を擬似コードによって示す。

【００７４】

【表５】

```
入力:   集約演算の個数: s
        各集約演算の結果の大きさの見積もり: n(1), n(2), ... n(s)
        システム全体の主記憶の大きさ: M
出力:   メモリに入る集約演算を区切って出力する.
アルゴリズム:
R = { 1, 2, ... s };            # 処理していない集約演算の集合
while R is not empty do
    m = M;                      # 残りのメモリサイズ
    Q = empty;                  # 結果が入る変数Qを空に初期化
    while m >:= 0 or R is empty do
        x = an element in R;    # R から一つ要素を取り出して
        R = R - { x };          # R から取り除き,
        Q = Q + { x };          # Q に加える.
        m = m - n(x);           # その要素が使用するメモリを引く.
    done;
    output(Q);              # Qをメモリに入る集約演算のグループとして
                            # 出力する.
done.
```

【００７５】（２）次に、どの集約演算のどのグループをどのプロセッサが実行するか決定する（ステップ５２０）。例えば、Ａという演算のグループとしてｂ，ｃ，ｄ，ｅが存在するならば、そのｂ，ｃ，ｄ，ｅをそれぞれプロセッサ１，２，３，４に割り当て、Ｂという演算のグループとしてｆ，ｇ，ｈ，ｉが存在するならば、そのｆ，ｇ，ｈ，ｉをそれぞれプロセッサ１，２，３，４に割り当てる等の処理を実施する。なお、各集約演算にどのようなグループが存在するかが予め分からないと本ステップは実行できないが、小規模のサンプリングを実行すれば、グループの種類を把握することができる。また、どのようにグループを割り当てるかは、様々な方法が考えられる。例えば、グループを入力とするハッシュ関数を用意しておき、ハッシュ値の範囲によってプロセッサを割り当てることも可能であるし、ユーザ指定により行うことも可能である。

【００７６】以上のステップ５１０及び５２０は、システム全体として実施する必要がある。よって、この処理は、特定のプロセッサを１つ割り当てて実施することも可能であるし、システムに制御用のプロセッサを設け、そのプロセッサに実行させることも可能である。

【００７７】（３）そして、各プロセッサはデータベースの自己の部分から今回処理する分を読み出す（ステップ５３０）。例えば、データベースの１タプルを読み出すようにすることも、複数のタプルを読み出すようにすることも可能である。

（４）各プロセッサは、読み出したデータのうち、実行する集約演算に必要なデータを選択する（ステップ５４０）。例えばデータベースの１タプルを読み出した場合、集約演算によっては不必要な属性についてのデータも読み出していることになるので、システム全体として不必要なデータについては破棄する。ここで、集約演算はステップ５１０で決定されたＴ個の集約演算であるから、このＴ個の集約演算に関係のないデータが破棄される。従来のＲｅｐアルゴリズムでは、１つの集約演算に関係するデータのみ用いられていたので、Ｒｅｐｍアルゴリズムではディスク装置への１回のアクセスが従来のＴ倍有効利用されていることが分かる。

【００７８】（５）次に、各プロセッサは、読み出したデータのうち他のプロセッサが集約すべきデータを、関係する集約演算のＩＤと共に当該他のプロセッサにネットワーク１を介して送信する（ステップ５５０）。実行する集約演算によって必要とするデータの属性が決定され、その属性値によりグループが決定されるので、この

属性値を入力とするハッシュ関数を用意し、その出力値により送信先のプロセッサが判別できるようにする。Ｔ個の集約演算の各々について本処理を実施する。この際、データを受信するプロセッサがどの集約演算についてのデータが送られてきたのが判別できるように集約演算のＩＤを付して送信する。

（６）各プロセッサは、そのプロセッサが集約すべき集約演算のグループに関するデータについてＴ個の集約演算を実行する（ステップ５６０）。この処理は、本ステップを実施するまでの演算結果と、読み出したデータのうち当該プロセッサが必要とするデータと、他のプロセッサから送られてきたデータとについて実施される。

【００７９】（７）そして、各プロセッサが、集約すべき集約演算のグループに関する全てのデータについてＴ個の集約演算を実行するまで、ステップ５３０乃至５６０を繰り返す（ステップ５７０）。すなわち、自己が読み出したデータ及び受信したデータが、全データベースの必要な部分となるまで繰り返される。

（８）もし、この処理が終了すると、Ｔ個の集約演算の処理が終了したことになる。よって、次のＴ個（これまでのＴとは異なる場合もある）の集約演算についてステップ５１０以下を、全ての（Ｓ個の）集約演算についての処理を実行するまで行う（ステップ５８０）。

【００８０】では、先に示した図７乃至図１０の例を用いて実行例を説明する。この際、同時に実行される集約演算は、先に示した演算１（月ごとの売上げ数を合計する演算）及び演算２（曜日ごとの売上げ数を合計する演算）とする。プロセッサ・エレメントは４つである。そして、プロセッサ・エレメント１は、演算１のＪａｎ及びＦｅｂのグループ、演算２のＭｏｎ及びＴｕｅのグループを集約する。プロセッサ・エレメント２は、演算１のＭａｒ及びＡｐｒのグループ、演算２のＷｅｄ及びＴｈｕのグループを集約する。プロセッサ・エレメント３は、演算１のＭａｙ及びＪｕｎのグループ、演算２のＦｒｉ及びＳａｔのグループを集約する。プロセッサ・エレメント４は、演算１のＪｕｌ及びＡｕｇのグループ、演算２のＳｕｎのグループを集約する。

【００８１】プロセッサ・エレメント３の処理を代表して説明する。図２５はプロセッサ・レメント３が図９に示したプロセッサ・エレメント３用のデータベースの第１行を読み出した状態を示す。この例では１行づつ読み出すが、一度に読み出す行数は任意である。本例では、日にち（ｄａｙ）の列、場所（ｌｏｃａｔｉｏｎ）の列、店番号（ｓｈｏｐ＃）の列及び製品番号（ｐｒｏｄｕｃｔ＃）の列は、演算１及び２に必要でないので破棄される（ステップ５４０参照）。一方、月（ｍｏｎｔｈ）の列と週（ｗｅｅｋ）の列と売上げ（ｓｏｌｄ）の列は演算１及び２に必要である。そして、第１行目の月"Ａｕｇ"及び売上げ"７"は、演算１のＩＤと共に集約を担当するプロセッサ・エレメント４に送信される（ステップ５５０参照）。また、第１行目の週"Ｓａｔ"及び売上げ"７"は、読み出したプロセッサ・エレメント３が行う演算２の集約すべきグループであるから、自ら集約を行う。ここでは最初のデータであるからそのまま記録する。プロセッサ・エレメント１及び２及び４も同時に、自己のデータベースの第１行目を読み出して処理を行っているとすると、プロセッサ・エレメント３には何も送られてこない。

【００８２】次に、プロセッサ・エレメント３が図９の第２行目を読み出した状態（不要な部分を除く）を図２６に示す。月"Ａｐｒ"及び売上げ"８"はプロセッサ・エレメント２に演算１のＩＤと共に送信される。また、週"Ｔｕｅ"及び売上げ"８"はプロセッサ・エレメント１に演算２のＩＤと共に送信される。プロセッサ・エレメント１及び２及び４も同時に、自己のデータベースの第２行目を読み出して処理を行っているとすると、プロセッサ・エレメント４から"Ｓａｔ"及び"９"が演算２のＩＤと共に送信されてくるので、プロセッサ・エレメント３は、第１行目の結果を参照して、"Ｓａｔ"及び"１６"を得る。

【００８３】プロセッサ・エレメント３が図９の第３行目を読み出した状態（不要な部分を除く）を図２７に示す。月"Ｍａｙ"及び売上げ"８"は、プロセッサ・エレメント３が集約すべき演算１のグループであるので、ここでは"Ｍａｙ"及び"８"をそのまま記録する。週"Ｔｈｕ"及び売上げ"８"は、プロセッサ・エレメント２に演算２のＩＤと共に送信される。プロセッサ・エレメント１及び２及び４も同時に、自己のデータベースの第３行目を読み出して処理を行っているとすると、プロセッサ・エレメント１からは演算１のＩＤと共に"Ｊｕｎ"及び"２"、プロセッサ・エレメント２からは演算１のＩＤと共に"Ｍａｙ"及び"２"が送られてくるので、プロセッサ・エレメント３はそれまでの結果を参照して、"Ｍａｙ"及び"１０"、"Ｊｕｎ"及び"２"を格納する。なお、ここでは、自らが読み出したデータと送信されてきたデータを別個に集約しているように説明しているが、これは説明の都合上であって、区別なく集約することができる。

【００８４】プロセッサ・エレメント３が図９の第４行目を読み出した状態（不要な部分を除く）を図２８に示す。月"Ｍａｒ"及び売上げ"８"は、プロセッサ・エレメント２が集約すべき演算１のグループであるので、プロセッサ・エレメント２に演算１のＩＤと共に送信する。また、週"Ｆｒｉ"及び売上げ"８"は、プロセッサ・エレメント３が集約すべき演算２のグループであるので、ここではそのまま格納する。プロセッサ・エレメント１及び２及び４も同時に、自己のデータベースの第４行目を読み出して処理を行っているとすると、プロセッサ・エレメント１からは演算１のＩＤと共に"Ｍａｙ"及び"４"と演算２のＩＤと共に"Ｆｒｉ"及び"４"が、プロセッサ・エレメント２からは演算２のＩＤと共に"Ｆｒｉ"及

び"９"が、プロセッサ・エレメント４からは演算１のＩＤと共に"Ｊｕｎ"及び"３"と演算２のＩＤと共に"Ｓｅｔ"及び"３"が送られてくる。よって、プロセッサ・エレメント３は、これらのデータから、"Ｍａｙ"及び"１４"と"Ｊｕｎ"及び"５"、"Ｆｒｉ"及び"２１"と"Ｓｅｔ"及び"１９"を計算する。

【００８５】以下これらの演算を繰り返すと、プロセッサ・エレメント３においては図１８の演算１及び演算２の部分が得られる。同様にして、プロセッサ・エレメント１においては図１６の演算１及び演算２の部分、プロセッサ・エレメント２においては図１７の演算１及び演算２の部分、プロセッサ・エレメント４においては図１９の演算１及び演算２の部分が計算されることとなる。なお、図２９に演算１のＩＤが付されて各プロセッサ・エレメントから他のプロセッサ・エレメントに送信されるデータの宛先をまとめておく。逆に、演算１について、各プロセッサ・エレメントに送られてくるデータの送信元を図３０にまとめておく。さらに、図３１に演算２のＩＤが付されて各プロセッサ・エレメントから他のプロセッサ・エレメントに送信されるデータの宛先をまとめておく。逆に、演算２について、各プロセッサ・エレメントに送られてくるデータの送信元を図３２にまとめておく。

【００８６】これまで説明してきたアルゴリズムをまとめてみる。２Ｐｍアルゴリズムは、各プロセッサ・エレメントが自己のディスク装置のデータに対して集約演算を実行するため、集約演算の最終結果と同じサイズのメモリを各プロセッサ・エレメントがそれぞれ用意しなければならない。しかし、通信コストが少ない分、最終結果が小さい場合には、他のアルゴリズムに比べて十分速い。

【００８７】ＢＣｍアルゴリズムは、集約演算の最終結果を全プロセッサ・エレメントのメモリで分担できるため２Ｐｍアルゴリズムが一度に処理できる集約演算のプロセッサ数倍だけ多くの集約演算に対してディスク装置へのアクセスを共有することができる。最終結果が１プロセッサ・エレメントのメモリに入りきらない場合、２Ｐｍアルゴリズムはメモリに入りきらない分を何度か繰り返し処理を行う必要があるが、ＢＣｍアルゴリズムではメモリ内に収まるようにできるため、一度に処理することができ、このような場合には２Ｐｍアルゴリズムより高速に処理することができる。

【００８８】最終結果を格納するのに必要なメモリサイズが非常に大きくなった場合、Ｒｅｐｍアルゴリズム以外のアルゴリズムはディスク装置へのアクセスや集計のコストが逆に大きくなるため、Ｒｅｐｍアルゴリズムが有利になる場合もある。

【００８９】よって、常に最適なアルゴリズムは同一ではないので、Ｒｅｐｍアルゴリズム、２Ｐｍアルゴリズム及びＢＣｍアルゴリズムのうち最適なアルゴリズムを選択し、そのアルゴリズムによって複数の集約演算を実行することが好ましい。そこで、各アルゴリズムを実行する際のコストを見積もることとする。以下、コストモデルについて説明する。

【００９０】最初に、生じ得る各コストを表６に掲げる。

【表６】

記　号	内　容	値
N	ノード数	可変
R	リレーションの大きさ（Byte）	5G
|R|	Rの行数	10000000
|Ri|	ノードiの行数	|R|／N
|Rp|	パーティション後の１ノード当りの行数	max(|Ri|, 1/S/Q)
P	ページ・サイズ	4KB
Q	集約演算の数	可変
p	プロジェクティビティ	0.5
S	１の集約演算のセレクティビティ	可変
S_f	２Pの第１フェーズのセレクティビティ	min(SN, 1)
Sg	２Pの第２フェーズのセレクティビティ	max(1／N, S)
|G|	結果の行数	|R|S
|Gi|	ノードiでの結果の行数	$\|Ri\|S_f$
M	ハッシュ・テーブルのサイズ	50M
IO	ページ読出しの時間	0.8ms
t_r	行読出しの時間	200/mips
t_w	行書込みの時間	200/mips
t_h	ハッシュ計算の時間	200/mips
t_a	集約計算の時間	40/mips
t_m	ページ通信の時間	0.1ms
t_b	ページ・ブロードキャストの時間	$(N-1)t_m$
mips	CPUのスピード	120/mips
T	ソースデータのスキャン回数	可変
A	１演算当りの集約ファンクションの数	100

なお、最も右側の列の値で、具体的数値が記載されている部分は一例であって、他のコンピュータ・システムでは他の値が用いられる。また、縦の線で囲まれた記号は、その数を示し、縦の線がない記号はそのバイト数を表す。

【００９１】幾つかの項目は説明を加えておく。ページ・サイズPは、コンピュータ・システムが取り扱うデータのサイズである。プロジェクティビティpは、リレーションの列のうち用いる割合を示す。１の集約演算のセレクティビティは、グループ数をデータ数で除した値を指す。また、同じ属性に注目する集約演算であっても属性の取扱いが異なる場合、例えば回数をカウントする場合もあれば合計を計算する場合、ある条件を満たす値のみ集約する場合等、もあるので、この異なる取扱いの数をAとする。データ・マイニングのような用途においては、大きな値の場合が多いので表６では１００という値にしている。

【００９２】以上の前提の下、各アルゴリズムのコストを示す。

【００９３】１．２Pmアルゴリズム

（ａ）データベースからの読み出しコスト

$(R_i/P) * IO * T$

（ｂ）必要なデータを選択するコスト

$|R_i| * t_r * T$

（ｃ）集約演算するコスト

$|R_i| * (t_h + t_a * A) * Q$

（ｄ）最初のハッシングで処理できなかった分のコスト

$(R_i * p * Q - M/S| * T) / P * 2 * IO$

（ｅ）結果を格納する行を生成するコスト

$|G_i| * t_w * Q$

（ｆ）送信受信コスト

$G_i/P * t_m * Q$

（ｇ）最終結果を計算するコスト

$|G_i| * (t_r + t_a * A) * Q$

（ｈ）結果を格納する行を生成するコスト（第２フェーズ分）

$|G_i| * Sg * t_w * Q$

（ｉ）最初のハッシングで処理できなかった分のコスト（第２フェーズ分）

$(G_i * Q - M/Sg * T) / P * 2 * IO$

（ｊ）結果をディスクに格納するコスト

$G_i * Sg/P * IO * Q$

これらのコストの合計が、２Pmアルゴリズムのコストということになる。

【００９４】２．BCmアルゴリズム

（ａ）データベースからの読み出しコスト

$(R_i/P) * IO * T$

（ｂ）ブロードキャストのコスト

$(R_i/P) * t_b * T$

（ｃ）通信バッファから行データを取り出すコスト

$|R_p| * t_r * T$

（ｄ）集約演算コスト

$|R| * (t_h + t_a * A) * Q$

（ｅ）最初のハッシングで処理できなかった分のコスト

（R_p＊ｐ＊Ｑ－Ｍ／Ｓ＊Ｔ）／Ｐ＊２＊ＩＯ
（ｆ）結果を格納する行を生成するコスト
｜R_p｜＊Ｓ＊t_w＊Ｑ
（ｈ）結果をディスク装置に格納するコスト
R_p＊Ｓ／Ｐ＊ｐ＊ＩＯ＊Ｑ
以上を合計すると、ＢＣｍアルゴリズムのコストが計算できる。
【００９５】３．Ｒｅｐｍアルゴリズム
（ａ）データベースからの読み出しコスト
（R_i／Ｐ）＊ＩＯ＊Ｔ
（ｂ）必要なデータを選択するためのコスト
｜R_i｜＊t_r＊Ｔ
（ｃ）宛先を探すためにハッシングし、通信バッファに書き込むためのコスト
｜R_i｜＊（t_h＋t_w）＊Ｑ
（ｄ）リパーティションしたものを送信／受信するコスト
R_p／Ｐ＊ｐ＊t_m＊Ｑ
（ｅ）集約演算コスト
｜R_p｜＊（t_r＋t_a＊Ａ）＊Ｑ
（ｆ）最初のハッシングで処理できなかった分のコスト
（R_p＊ｐ＊Ｑ－Ｍ／Ｓ＊Ｔ）／Ｐ＊２＊ＩＯ
（ｇ）結果を格納する行を生成するコスト
｜R_p｜＊Ｓ＊t_w＊Ｑ
（ｈ）結果をディスク装置に格納するコスト
R_p＊Ｓ／Ｐ＊ｐ＊ＩＯ＊Ｑ
以上を合計することにより、Ｒｅｐｍアルゴリズムに必要なコストを計算することができる。
【００９６】このように各アルゴリズムの実行に必要なコストを計算することができる。これを用いて、以下のような処理（図３３）にて実行すべきアルゴリズムを決定する。最初に、ハードウエアに関するパラメータを入力する（ステップ５１０）。このハードウエアに関するパラメータは、表６のプロセッサ・エレメントの数、ハッシュ・テーブルのサイズ、ページ読み出しの時間、行読み出しの時間、行書き込みの時間、ハッシュ計算の時間、ページ送信の時間、集約演算の時間、ページブロードキャストの時間等、ハードウエア構成から決定するパラメータである。これらのパラメータは、一度コンピュータの構成が決まると変わらないものが多いが、計算によってはプロセッサ・エレメントの数を変更したり、ハッシュ・テーブルのサイズも変更可能である。次に実行する集約演算に関するパラメータを入力する（ステップ５２０）。表６の項目のうちハードウエアのパラメータとしてもの以外は、ほぼ集約演算に関するパラメータである。結果の行数等は集約演算のみでは決定しないが、サンプリングを行いグループ数等を見積もることで数値を得ることができる。このような事項は、従来から行われてきたことである（例えば、P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stockes, "Sampling-based estimation of the number of distinct values of an attribute," In Proceedings of the 21st VLDB Conference, pages 311-322, 1995などを参照のこと）。
【００９７】このように表６に記されたパラメータが入力されると、先に示したコストモデルにて各アルゴリズムのコストを計算することができる（ステップ５３０）。その後、その３つのアルゴリズムのうち最小コストのアルゴリズムを決定する（ステップ５４０）。これにより、決定されたアルゴリズムにて集約演算を実行すれば、３つのアルゴリズムのうち最も高速に演算を実行することができる。
【００９８】以上、本発明を実施するために必要とされるコンピュータ・システム及びプログラムに処理フローについて説明した。プログラムの処理フローの各ステップを実行するような回路や、プログラムと回路の組み合わせによって、各ステップを実現する手段を形成することも可能である。各ステップを実現する手段は、各プロセッサ・エレメントの代わりに設け、高速のネットワークで並列処理する。繰り返しを制御するような制御回路を設けること、装置全体を管理するような管理手段を設けることも考えられる。
【００９９】また、以上の処理フローを実現するためのプログラムは、フロッピー・ディスク、ＣＤ－ＲＯＭやＭＯ（Magneto-Optics）ディスク等の記憶媒体上に実現され、流通することが考えられる。さらに、記憶媒体のみならず、有線や無線の通信媒体上で流通する場合もある。このような場合、上述した処理フローの各ステップのうち、幾つかのステップのためプログラムは、記憶媒体、通信媒体にて流通するプログラムとは別途ユーザに提供される場合も考えられる。
【０１００】
【実施例】図３４は、１６ノードのハイパフォーマンス・スイッチを伴うＩＢＭ　ＳＰ２を用いた場合に、どのアルゴリズムを用いるとよいかを示すものである。このシステムの場合、領域ａは２Ｐｍアルゴリズムが最適で、領域ｂはＢＣｍアルゴリズムが最適で、領域ｃはＲｅｐｍアルゴリズムが最適である。演算数と１演算当たりのグループ数が分かれば、最適なアルゴリズムを選択することができる。なお、このような図を作成して最適アルゴリズムを選択する訳ではなく、先に示したコストモデルで各アルゴリズムのコストを計算し、そのコストとの対比にて最適アルゴリズムを選択する。
【０１０１】
【効果】複数の集約演算を並列に高速実行する方法を提供することができた。
【０１０２】上記の方法を用いると複数の集約演算を並列に高速実行できるので、ＯＬＡＰやデータ・マイニングの実行を高速にすることもできる。
【０１０３】さらに、複数の集約演算を実行する方法を、ハードウエアの条件や複数の集約演算の性質等に基

づき、切り換えることができるようにし、様々な条件の下、複数の集約演算を常に同じ方法で実行するよりも、より高速に実行することができた。

【図面の簡単な説明】

【図１】Ｒｅｐアルゴリズムの処理フローを示した図である。

【図２】２Ｐアルゴリズム、Ｒｅｐアルゴリズム、及びＢＣアルゴリズムにおける、グループ数対応答時間の関係を示した図である。

【図３】本発明で用いられるコンピュータ・システムの構成の一例である。

【図４】本発明で用いられるコンピュータ・システムの構成の一例である。

【図５】本発明で用いられるコンピュータ・システムの構成の一例である。

【図６】２Ｐｍアルゴリズムの処理フローを示した図である。

【図７】プロセッサ・エレメント１のデータベースの内容を表す図である。

【図８】プロセッサ・エレメント２のデータベースの内容を表す図である。

【図９】プロセッサ・エレメント３のデータベースの内容を表す図である。

【図１０】プロセッサ・エレメント４のデータベースの内容を表す図である。

【図１１】２Ｐｍアルゴリズムの２番目のステップを図７に対して実行中のプロセッサ・エレメント１のメモリ内の途中結果を示す図である。

【図１２】２Ｐｍアルゴリズムの２番目のステップを図７に対して実行したプロセッサ・エレメント１のメモリ内の結果を示す図である。

【図１３】２Ｐｍアルゴリズムの２番目のステップを図８に対して実行したプロセッサ・エレメント２のメモリ内の結果を示す図である。

【図１４】２Ｐｍアルゴリズムの２番目のステップを図９に対して実行したプロセッサ・エレメント３のメモリ内の結果を示す図である。

【図１５】２Ｐｍアルゴリズムの２番目のステップを図１０に対して実行したプロセッサ・エレメント４のメモリ内の結果を示す図である。

【図１６】２Ｐｍアルゴリズムの集計処理を実行したプロセッサ・エレメント１のメモリ内の結果を示す図である。

【図１７】２Ｐｍアルゴリズムの集計処理を実行したプロセッサ・エレメント２のメモリ内の結果を示す図である。

【図１８】２Ｐｍアルゴリズムの集計処理を実行したプロセッサ・エレメント３のメモリ内の結果を示す図である。

【図１９】２Ｐｍアルゴリズムの集計処理を実行したプロセッサ・エレメント４のメモリ内の結果を示す図である。

【図２０】ＢＣｍアルゴリズムの処理フローを示した図である。

【図２１】ＢＣｍアルゴリズムでどのプロセッサ・エレメントがどの集約演算を実行するのかを決定するための処理フローを示す図である。

【図２２】図７乃至図１０に対し、ＢＣｍアルゴリズムを実行したプロセッサ・エレメント２のメモリ内の結果を示す図である。

【図２３】ＢＣｍアルゴリズムを実行したプロセッサ・エレメント１．３乃至４のメモリ内の結果を示す図である。

【図２４】Ｒｅｐｍアルゴリズムの処理フローを示した図である。

【図２５】Ｒｅｐｍアルゴリズムで演算１及び演算２を実行するプロセッサ・エレメント３の処理の途中の状況を説明するための図である。

【図２６】Ｒｅｐｍアルゴリズムで演算１及び演算２を実行するプロセッサ・エレメント３の処理の途中の状況を説明するための図である。

【図２７】Ｒｅｐｍアルゴリズムで演算１及び演算２を実行するプロセッサ・エレメント３の処理の途中の状況を説明するための図である。

【図２８】Ｒｅｐｍアルゴリズムで演算１及び演算２を実行するプロセッサ・エレメント３の処理の途中の状況を説明するための図である。

【図２９】Ｒｅｐｍアルゴリズムで演算１及び演算２を実行する際に各プロセッサ・エレメントにおいて送信される、演算１関連のデータ及びその送信先を表す図である。

【図３０】Ｒｅｐｍアルゴリズムで演算１及び演算２を実行する際に各プロセッサ・エレメントにおいて受信される、演算１関連のデータ及びその送信元を表す図である。

【図３１】Ｒｅｐｍアルゴリズムで演算１及び演算２を実行する際に各プロセッサ・エレメントにおいて送信される、演算２関連のデータ及びその送信先を表す図である。

【図３２】Ｒｅｐｍアルゴリズムで演算１及び演算２を実行する際に各プロセッサ・エレメントにおいて受信される、演算２関連のデータ及びその送信元を表す図である。

【図３３】２Ｐｍ，ＢＣｍ，Ｒｅｐｍアルゴリズムのうち、最適なアルゴリズムを選択するための処理フローを示す図である。

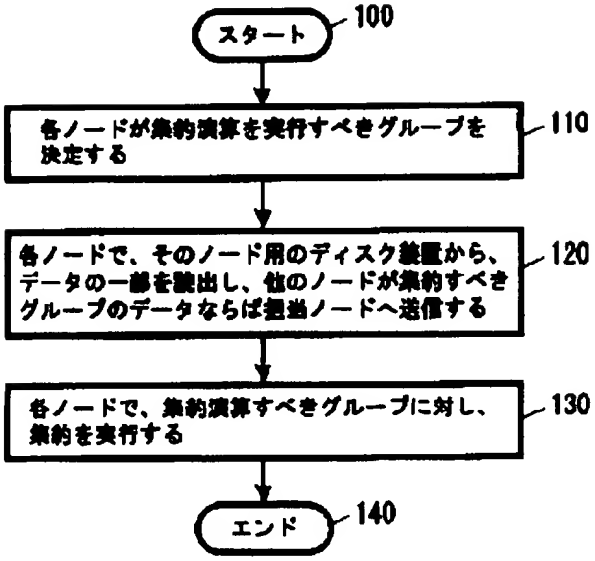
【図３４】１６ノードのＩＢＭ　ＳＰ２で集約演算を実行する際のアルゴリズム切り換え位置を示す図である。

【符号の説明】

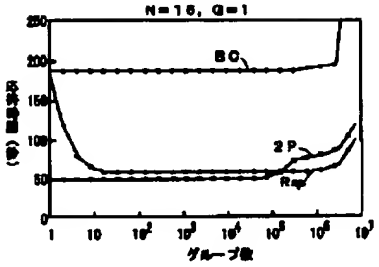
１　ネットワーク

３ａ，３ｂ，３ｃ　プロセッサ・エレメント
５ａ，５ｂ，５ｃ　メモリ
７ａ，７ｂ，７ｃ　ディスク装置
９　メモリ
９ａ，９ｂ　メモリ領域
１０　メモリ
１１　ディスク装置
１１ａ，１１ｂ　ディスク領域
１３　ディスク装置

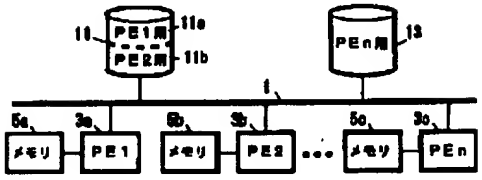
【図１】



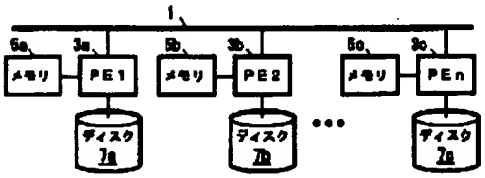
【図２】



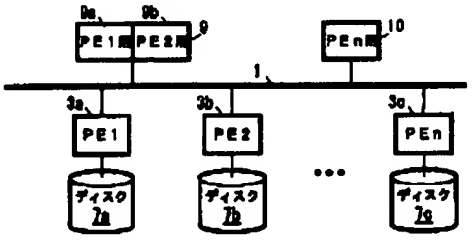
【図５】



【図３】



【図４】



【図７】

【図８】

【図２２】

ノード１

month	day	week	location	shop#	product#	sold
Apr	7	Sun	Kyushu	8	2	3
Aug	4	Sun	Hokkaido	12	1	4
Jun	17	Mon	Hokkaido	15	1	2
May	10	Fri	Kansai	12	1	4
Mar	13	Wed	Kansai	6	1	4
Jul	28	Sun	Kyushu	14	3	1
Apr	25	Tue	Kansai	16	1	9
Mar	15	Sat	Kansai	13	1	6
Mar	15	Sat	Hokkaido	7	2	6

ノード２

month	day	week	location	shop#	product#	sold
Jul	2	Wed	Kansai	11	2	6
Mar	31	Tue	Kyushu	5	3	1
May	6	Mon	Kanto	1	2	2
Aug	2	Fri	Hokkaido	6	1	9
May	3	Thu	Kansai	6	3	7
Apr	23	Mon	Kyushu	16	2	3
Mar	26	Tue	Kanto	14	3	6
Jul	20	Sat	Kanto	17	3	2
Aug	18	Sun	Kanto	11	1	9

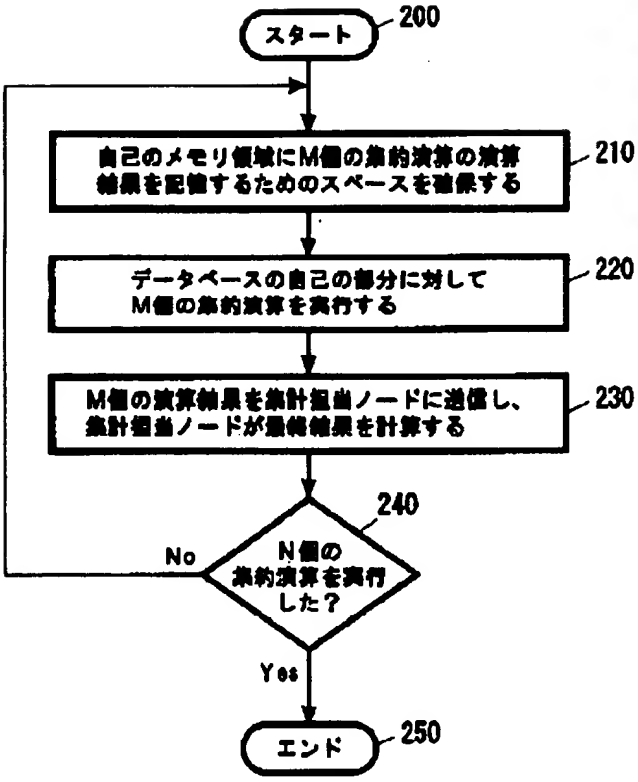
演算２

week	sum(sold)
Mon	7
Tue	28
Wed	28
Thu	26
Fri	27
Sat	43
Sun	35

演算３

location	sum(sold)
Hokkaido	58
Kanto	61
Kansai	52
Kyushu	28

【図6】



【図9】

ノード3

month	day	week	location	dept#	product#	sold
Aug	3	Sat	Kyushu	8	2	7
Apr	8	Tue	Hokkaido	10	1	5
May	6	Thu	Hokkaido	15	1	6
Mar	1	Fri	Kanto	20	9	6
Apr	30	Tue	Hokkaido	12	1	5
Mar	5	Tue	Hokkaido	1	1	5
Apr	11	Thu	Kansai	5	2	1
Apr	0	Sat	Kanto	5	1	9
Mar	27	Wed	Hokkaido	5	2	1

【図16】

演算1

month	sum(sold)
Jan	4
Feb	8

演算2

week	sum(sold)
Mon	7
Tue	35

演算3

location	sum(sold)
Hokkaido	88

演算4

product#	sum(sold)
1	186

演算5

location	week	sum(sold)
Hokkaido	Mon	2
Hokkaido	Tue	17
Hokkaido	Wed	10
Hokkaido	Thu	5
Hokkaido	Fri	9
Hokkaido	Sat	8
Hokkaido	Sun	4

【図10】

ノード4

month	day	week	location	dept#	product#	sold
Jul	10	Wed	Hokkaido	8	1	9
Aug	12	Sat	Kanto	12	3	8
Apr	21	Sun	Kanto	15	3	6
Jun	20	Sat	Kansai	14	2	3
Aug	15	Fri	Kansai	1	3	6
Feb	6	Wed	Kanto	13	1	5
Jan	19	Sat	Kansai	6	3	4
Mar	19	Thu	Hokkaido	13	1	3
Mar	1	Sat	Kyushuu	3	1	5

【図11】

演算1

month	sum(sold)
Apr	3
Jun	2
Aug	4

演算2

week	sum(sold)
Mon	1
Sun	7

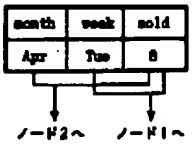
演算3

location	sum(sold)
Hokkaido	6
kyusuu	3

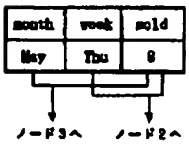
演算4

product#	sum(sold)
1	6
2	3

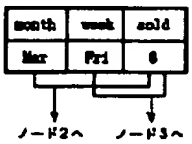
【図26】



【図27】



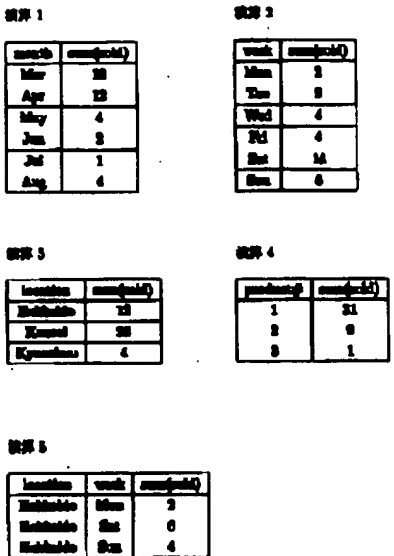
【図28】



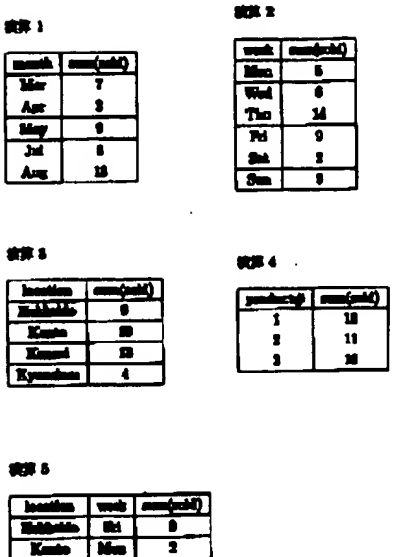
演算5

location	week	sum(sold)
Hokkaido	Sun	4
Hokkaido	Mon	2
kyusuu	Sun	3

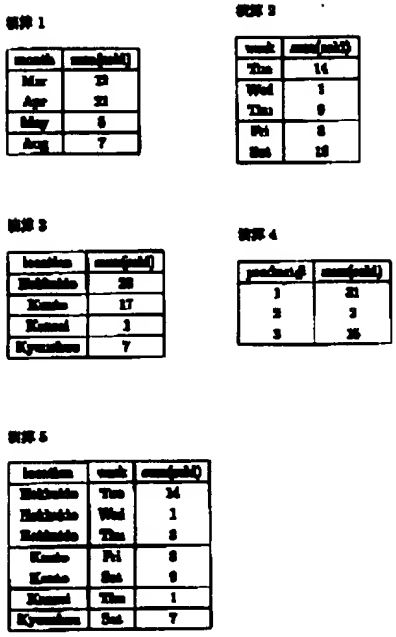
【図12】



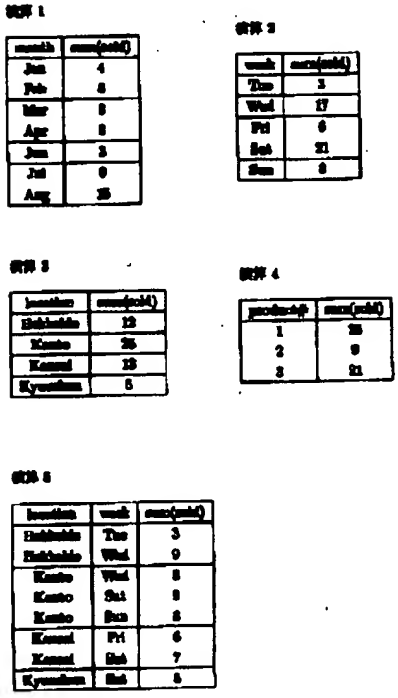
【図13】



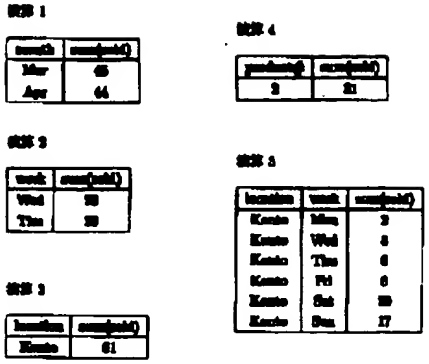
【図14】



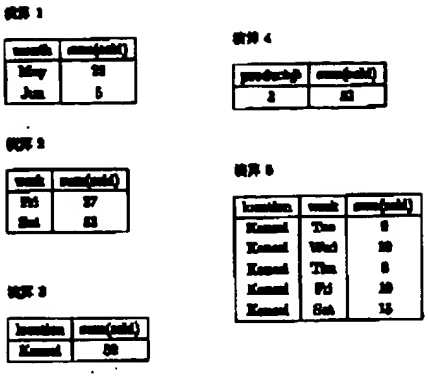
【図15】



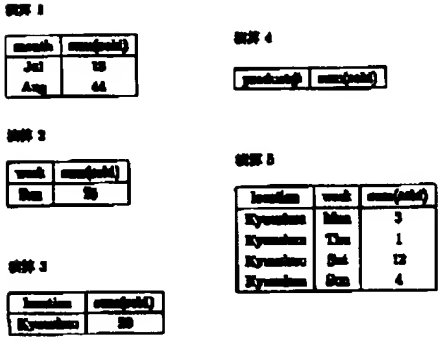
【図17】



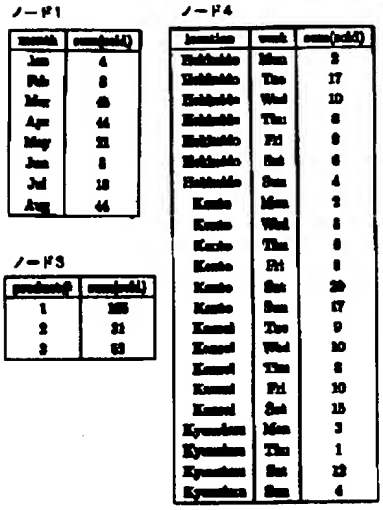
【図18】



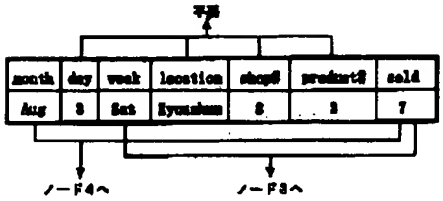
【図19】



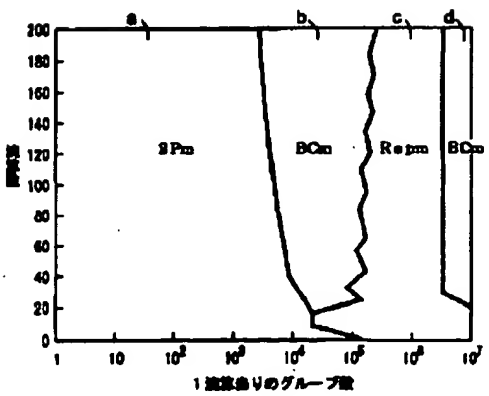
【図23】



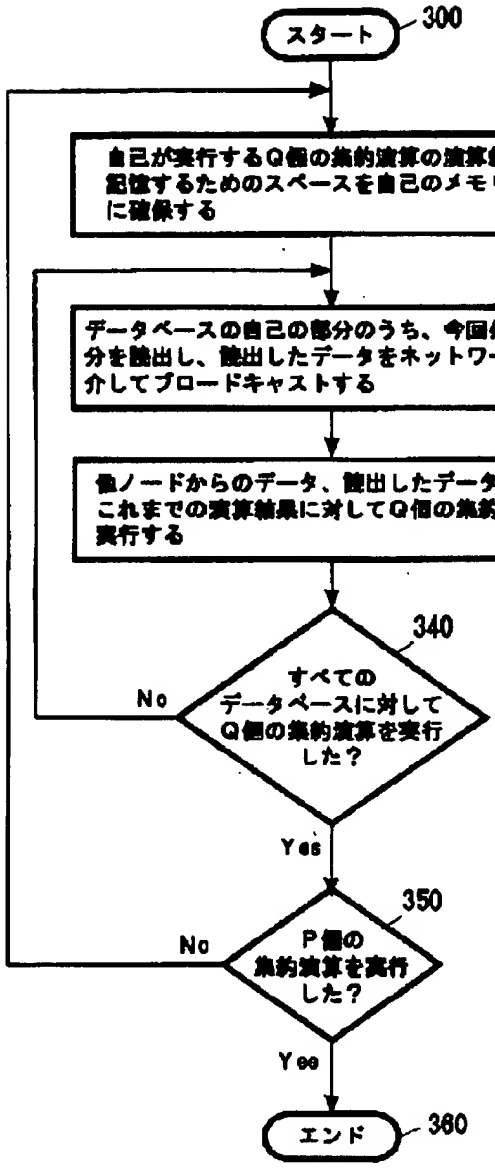
【図25】



【図34】



【図20】



【図29】

ノードF1

month	sold	
Apr	3	⇒ノードF2
Aug	4	⇒ノードF4
Jun	2	⇒ノードF3
May	4	⇒ノードF3
Mar	4	⇒ノードF2
Jul	1	⇒ノードF4
Apr	9	⇒ノードF2
Mar	8	⇒ノードF2
Mar	6	⇒ノードF2

ノードF2

month	sold	
Jul	6	⇒ノードF4
Mar	1	⇒ノードF2
May	2	⇒ノードF3
Aug	8	⇒ノードF4
May	7	⇒ノードF3
Apr	3	⇒ノードF2
Mar	6	⇒ノードF2
Jul	2	⇒ノードF4
Aug	8	⇒ノードF4

ノードF3

month	sold	
Aug	7	⇒ノードF4
Apr	8	⇒ノードF2
May	6	⇒ノードF3
Mar	6	⇒ノードF2
Apr	3	⇒ノードF2
Mar	3	⇒ノードF2
Apr	1	⇒ノードF2
Apr	9	⇒ノードF2
Mar	1	⇒ノードF2

ノードF4

month	sold	
Jul	9	⇒ノードF4
Aug	9	⇒ノードF4
Apr	8	⇒ノードF2
Jun	8	⇒ノードF3
Aug	6	⇒ノードF4
Feb	8	⇒ノードF1
Jan	4	⇒ノードF1
Mar	3	⇒ノードF2
Mar	5	⇒ノードF2

【図31】

ノードF1

month	sold	
Sun	3	⇒ノードF4
Sun	4	⇒ノードF4
Mon	2	⇒ノードF1
Fri	4	⇒ノードF3
Wed	4	⇒ノードF2
Sun	1	⇒ノードF4
Tue	9	⇒ノードF1
Sat	8	⇒ノードF3
Sat	6	⇒ノードF3

ノードF2

month	sold	
Wed	6	⇒ノードF2
Thu	1	⇒ノードF2
Mon	2	⇒ノードF1
Fri	9	⇒ノードF3
Thu	7	⇒ノードF2
Mon	3	⇒ノードF1
Thu	6	⇒ノードF2
Sat	2	⇒ノードF3
Sun	9	⇒ノードF4

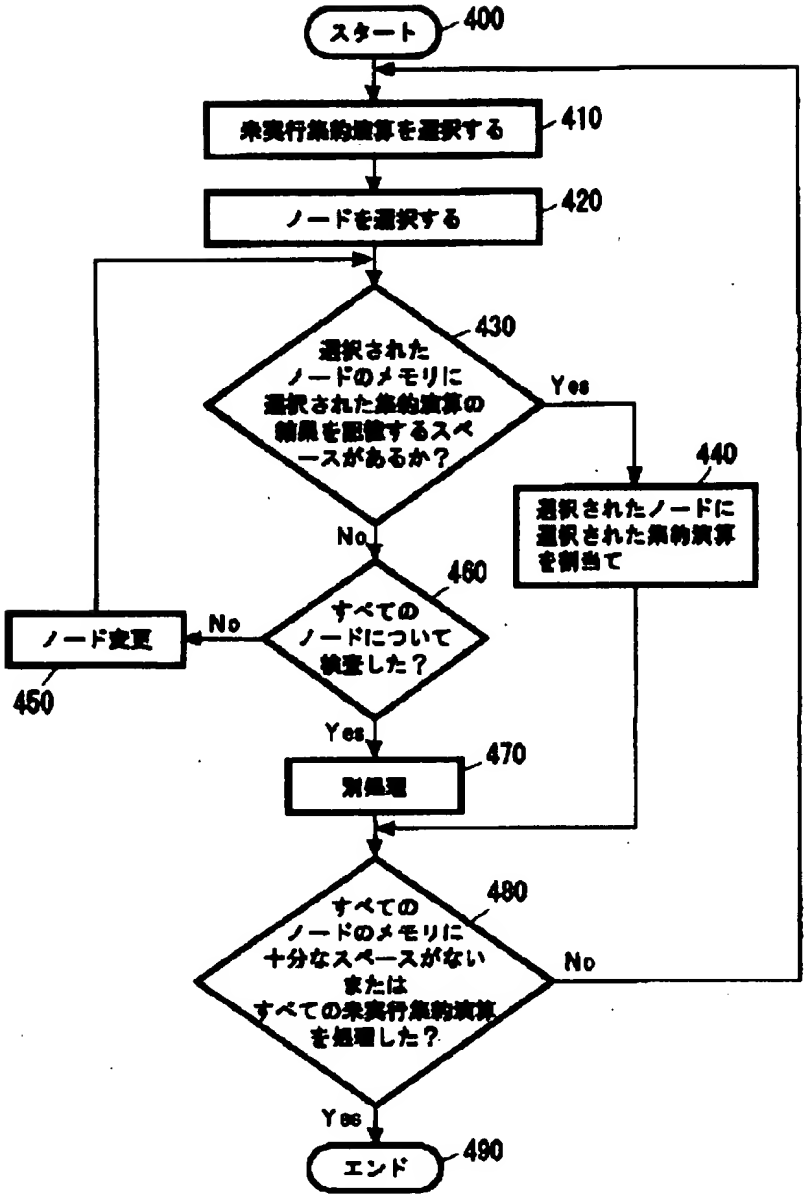
ノードF3

month	sold	
Sat	7	⇒ノードF3
Tue	8	⇒ノードF1
Thu	8	⇒ノードF2
Fri	8	⇒ノードF3
Tue	8	⇒ノードF1
Tue	3	⇒ノードF1
Thu	1	⇒ノードF2
Sat	9	⇒ノードF3
Wed	1	⇒ノードF2

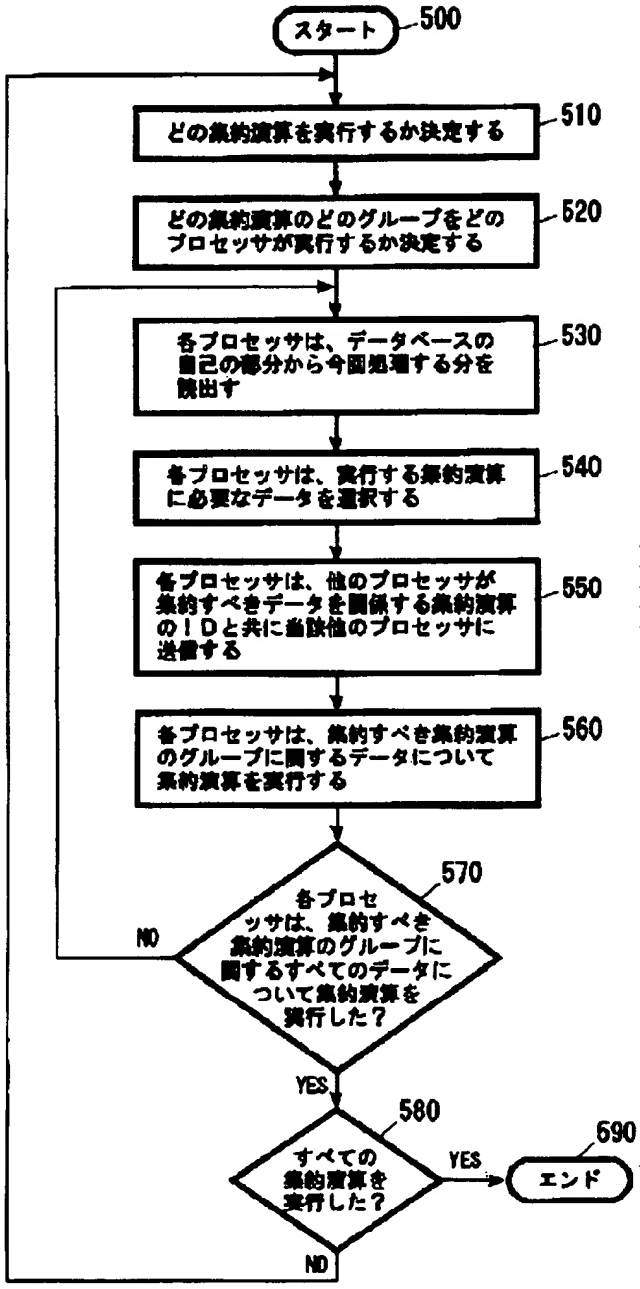
ノードF4

month	sold	
Wed	9	⇒ノードF2
Sat	9	⇒ノードF3
Sun	8	⇒ノードF4
Sat	8	⇒ノードF3
Fri	6	⇒ノードF3
Wed	8	⇒ノードF2
Sat	4	⇒ノードF3
Tue	3	⇒ノードF1
Sat	5	⇒ノードF3

【図21】


スタート
400
非実行集約演算を選択する
410
ノードを選択する
420
選択された
ノードのメモリに
選択された集約演算の
結果を記憶するスペ
ースがあるか？
430
Yes
No
選択されたノードに
選択された集約演算
を割当て
440
すべての
ノードについて
検査した？
460
No
ノード変更
450
Yes
別処理
470
すべての
ノードのメモリに
十分なスペースがない
または
すべての非実行集約演算
を処理した？
480
No
Yes
エンド
490

【図２４】



【図３０】

ノードＦ１

	month	sold
ノードＦ４ ⇒	Feb	8
ノードＦ４ ⇒	Jan	4

ノードＦ２

	month	sold
ノードＦ１ ⇒	Apr	3
ノードＦ１ ⇒	Mar	4
ノードＦ１ ⇒	Apr	9
ノードＦ１ ⇒	Mar	8
ノードＦ１ ⇒	Mar	6
ノードＦ２ ⇒	Mar	1
ノードＦ２ ⇒	Apr	3
ノードＦ２ ⇒	Mar	6
ノードＦ３ ⇒	Apr	8
ノードＦ３ ⇒	Mar	8
ノードＦ３ ⇒	Apr	3
ノードＦ３ ⇒	Mar	3
ノードＦ３ ⇒	Apr	1
ノードＦ３ ⇒	Apr	9
ノードＦ３ ⇒	Mar	1
ノードＦ４ ⇒	Apr	8
ノードＦ４ ⇒	Mar	3
ノードＦ４ ⇒	Mar	5

ノードＦ３

	month	sold
ノードＦ１ ⇒	Jun	2
ノードＦ１ ⇒	May	4
ノードＦ２ ⇒	May	2
ノードＦ２ ⇒	May	7
ノードＦ３ ⇒	May	8
ノードＦ４ ⇒	Jun	3

ノードＦ４

	month	sold
ノードＦ１ ⇒	Aug	4
ノードＦ１ ⇒	Jul	1
ノードＦ２ ⇒	Jul	6
ノードＦ２ ⇒	Aug	9
ノードＦ２ ⇒	Jul	2
ノードＦ２ ⇒	Aug	9
ノードＦ３ ⇒	Aug	7
ノードＦ４ ⇒	Jul	9
ノードＦ４ ⇒	Aug	9
ノードＦ４ ⇒	Aug	6

【図３２】

ノードＦ１

	month	sold
ノードＦ１ ⇒	Mon	2
ノードＦ１ ⇒	Tue	9
ノードＦ２ ⇒	Mon	2
ノードＦ２ ⇒	Mon	3
ノードＦ３ ⇒	Tue	6
ノードＦ３ ⇒	Tue	3
ノードＦ３ ⇒	Tue	3
ノードＦ４ ⇒	Tue	3

ノードＦ２

	month	sold
ノードＦ１ ⇒	Wed	4
ノードＦ２ ⇒	Wed	6
ノードＦ３ ⇒	Thu	1
ノードＦ３ ⇒	Thu	7
ノードＦ３ ⇒	Thu	6
ノードＦ３ ⇒	Thu	8
ノードＦ３ ⇒	Thu	1
ノードＦ３ ⇒	Wed	1
ノードＦ４ ⇒	Wed	9
ノードＦ４ ⇒	Wed	8

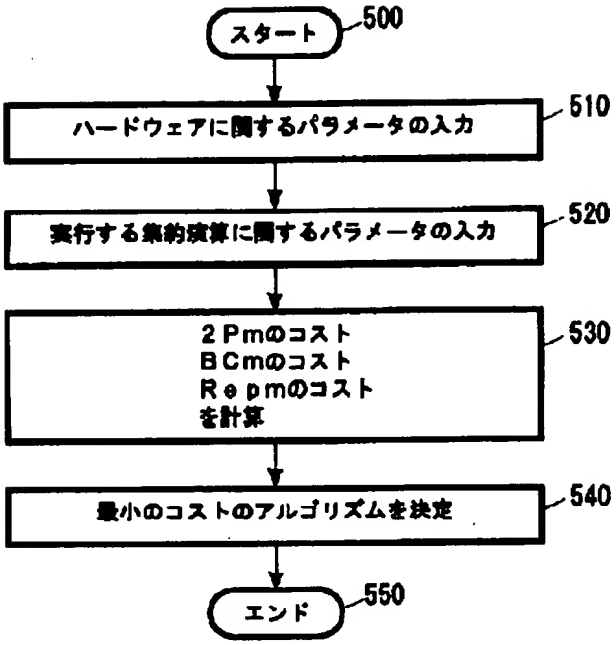
ノードＦ３

	month	sold
ノードＦ１ ⇒	Fri	4
ノードＦ１ ⇒	Sat	6
ノードＦ１ ⇒	Sat	6
ノードＦ２ ⇒	Fri	9
ノードＦ２ ⇒	Sat	2
ノードＦ３ ⇒	Sat	7
ノードＦ３ ⇒	Fri	8
ノードＦ３ ⇒	Sat	9
ノードＦ４ ⇒	Sat	9
ノードＦ４ ⇒	Sat	8
ノードＦ４ ⇒	Fri	6
ノードＦ４ ⇒	Sat	4
ノードＦ４ ⇒	Sat	5

ノードＦ４

	month	sold
ノードＦ１ ⇒	Sun	8
ノードＦ１ ⇒	Sun	4
ノードＦ１ ⇒	Sun	1
ノードＦ２ ⇒	Sun	9
ノードＦ４ ⇒	Sun	8

【図33】


スタート
500
ハードウェアに関するパラメータの入力
510
実行する集約演算に関するパラメータの入力
520
2Pmのコスト
BCmのコスト
Repmのコスト
を計算
530
最小のコストのアルゴリズムを決定
540
エンド
550